OBJECT COUNTING WITH DEEP LEARNING

A Thesis Submitted to the
College of Graduate and Postdoctoral Studies
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Shubhra Aich

PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Or

Dean
College of Graduate and Postdoctoral Studies
University of Saskatchewan
116 Thorvaldson Building, 110 Science Place
Saskatoon, Saskatchewan S7N 5C9
Canada

ABSTRACT

This thesis explores various empirical aspects of deep learning or convolutional network based models for efficient object counting. First, we train moderately large convolutional networks on comparatively smaller datasets containing few hundred samples from scratch with conventional image processing based data augmentation. Then, we extend this approach for unconstrained, outdoor images using more advanced architectural concepts. Additionally, we propose an efficient, randomized data augmentation strategy based on sub-regional pixel distribution for low-resolution images.

Next, the effectiveness of depth-to-space shuffling of feature elements for efficient segmentation is investigated for simpler problems like binary segmentation – often required in the counting framework. This depth-to-space operation violates the basic assumption of encoder-decoder type of segmentation architectures. Consequently, it helps to train the encoder model as a sparsely connected graph. Nonetheless, we have found comparable accuracy to that of the standard encoder-decoder architectures with our depth-to-space models.

After that, the subtleties regarding the lack of localization information in the conventional scalar count loss for one-look models are illustrated. At this point, without using additional annotations, a possible solution is proposed based on the regulation of a network-generated heatmap in the form of a weak, subsidiary loss. The models trained with this auxiliary loss alongside the conventional loss perform much better compared to their baseline counterparts, both qualitatively and quantitatively. Lastly, the intricacies of tiled prediction for high-resolution images are studied in detail, and a simple and effective trick of eliminating the normalization factor in an existing computational block is demonstrated. All of the approaches employed here are thoroughly benchmarked across multiple heterogeneous datasets for object counting against previous, state-of-the-art approaches.

ACKNOWLEDGEMENTS

I would like to thank Dr. Ian Kent Stavness, whose kind supervision with generous computational support made this work possible. I extend my gratitude to all the co-workers under P^2IRC theme 3.2, especially to William van der Kamp, Ilya Ovsyannikov, Anique Josuttes, and Keegan Strueby for painstakingly organize the necessary data for several experiments in this work. Also, I must express my heartfelt gratitude to Gwen Lancaster, who is largely responsible to make my life (and possibly, many other students') smoother in my early days of burning winter in Saskatoon and afterward.

This research was undertaken thanks in part to funding from the Canada First Research Excellence Fund, the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the Microsoft AI for Earth and NVIDIA GPU programs.

Dedicated to my wife

CONTENTS

LIST OF TABLES

LIST OF FIGURES

STATEMENT OF AUTHORSHIP

The chapters in this thesis contains the work published or submitted in the conferences, workshops, and journals listed below:

- Chapter 2: Leaf counting with deep convolutional and deconvolutional networks
  S. Aich and I. Stavness. *Proceedings of the 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), Venice, Italy.*
  This paper ranked 2^{nd} place in the Leaf Counting Competition (LCC2017), and got best poster award at the CVPPP workshop.

- Chapter 3: Deepwheat: Estimating phenotypic traits from crop images with deep learning
  S. Aich, A. Josuttes, I. Ovsyannikov, K. Strueby, I. Ahmed, H. S. Duddu, C. Pozniak, S. Shirtliffe, and I. Stavness. *2018 IEEE Winter Conference on Applications of Computer Vision (WACV).*

- Chapter 4: Semantic Binary Segmentation using Convolutional Networks without Decoders
  S. Aich, W. van der Kamp, and I. Stavness. *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW).*
  This paper ranked 5^{th} place in the Road Segmentation Challenge at the DeepGlobe workshop.

- Chapter 5: Preventing False Localization in One-Look Object Counting Models
  S. Aich and I. Stavness. *Under review in Pattern Recognition Letters (Elsevier).*
  The preliminary version of this work is also available in arxiv.org as follows:
  Improving Object Counting with Heatmap Regulation. S. Aich and I. Stavness. *arXiv:1803.05494.*

- Chapter 6: Object Counting with Small Datasets of Large Images
  S. Aich and I. Stavness. *Under review in 2019 IEEE Conference on Computer Vision and Pattern Recognition Workshops.*

All the computational aspects reported in these papers, from data augmentation to architectural design, implementation, and training are accomplished by me under the supervision of Dr. Ian Stavness. Rest of the authors mostly contributed to the collection of data and preparation of manuscripts.

1. INTRODUCTION

1.1 Problem Definition

Counting object instances from images and videos is a common and practical computer vision task found in a range of applications, such as counting vehicles from aerial images [46, 70, 76], crowd counting for surveillance [22, 60, 87, 100, 117], biological cell counting for medical diagnosis [60, 111, 112], plant counting for image based plant phenotyping [37, 74, 82, 84, 89, 106], and so on. From a broader, more theoretical perspective, object counting can be categorized as a sub-domain of object detection which is a subspace of instance-level segmentation. In this regard, solving the problem of instance segmentation is sufficient (not necessary) to provide solutions for both object detection and counting. However, instance segmentation [41] and object detection [81, 83] pipelines demand annotation images with much higher specificity as compared to mere object counting frameworks. Obtaining large-scale annotated datasets with fine granularity is a prohibitively time-consuming process. Thus, specialized and computationally efficient counting approaches that exhibit similar performance with weaker image labels (i.e., dot annotations as compared to bounding boxes or pixel-level masks) are worth pursuing for real-time autonomous vision systems where an object count alone is needed.

In this thesis, we investigate the idea of deep learning, more specificically convolutional neural networks (ConvNets), for object counting from visual data acquired from various sources. In the realm of deep learning, two different kinds of approaches have gained popularity as the possible solution for counting problems:

- The older one, density map estimation followed by additional post-processing, has its root in the conventional feature extraction with classification or regression based frameworks [12, 33, 60]. With the recent emergence of deep learning, the old-fashioned, constant-dimensional feature extraction based density estimators have been replaced by convolutional networks [13, 26, 77, 91, 111, 112]. Nonetheless, post-processing steps are needed to retrieve the final count resulting in a multi-stage process. Moreover, these kind of density estimation networks belong to the family of encoder-decoder type architectures, incurring additional computation-cost due to the extra decoder sub-network compared to simple encoder-only type models.

- The later option that evolved recently, one-look regression models [46, 70], are the encoder-only convolutional networks. They are very similar to the classification architectures [44, 55, 98] with the only difference that in regression models, there are either a single output unit responsible for generating the scalar count [46] or the number of output units is a reasonable *supremum* of the number of objects in a single image [70]. Regardless of the choice of output layer, these one-look models are both end-to-end and computationally much more efficient compared to their

density estimation based counterparts.

In this thesis, we have chosen the later path of one-look models for exploration due to its comparative efficiency and simplicity in design. We investigate various constraints involved in employing regarding one-look models for object counting in general, such as small datasets and data augmentation, limitations of the standard loss formulations, efficient foreground segmentation, and variable input resolution. In this endeavor, we propose a few useful tricks to adapt the existing one-look models or ConvNet designs to build efficient, general-purpose object counting systems. Although the later chapters follow a progression towards the idea of building a generalized object counting pipeline, these chapters are either published or under submission as standalone manuscripts, and so, can be read independently. Below we provide a brief account of the contributions inscribed in each chapter.

1.2 Contributions

- **Chapter 2:** In the domain of object counting, our journey started through the participation into the CVPPP 2017 Leaf Counting Challenge (LCC), where we obtained reasonable accuracy by training a single, moderately large, VGG-style ConvNet to count leaves from top-down images of rosette plants. Our approach used segmented leaf images from all categories, and was trained from scratch with data augmentation based on simple image processing. Before us, the recent work on the similar dataset used tiny, customized models, one for each category of images. Moreover, the other competitors finetuned very large, pre-trained models on the small-scale competition dataset to achieve state-of-the-art performance. In this regard, we were the first to train a considerably large ConvNet from scratch with few hundred training samples for homogeneous object counting without significant overfitting. The full story of this implementation is depicted in Chapter 2.

- **Chapter 3:** The datasets used in Chapter 2 comprise images acquired in highly controlled environment. At this point, our curiosity drove us to explore similar techniques developed in the previous chapter for images captured in uncontrolled, outdoor environments. Henceforth, we conducted similar experiments for two plant datasets. The first one is for counting the number of early-season wheat plants in sufficiently high-resolution images captured with a GoPro camera. The second dataset is for estimating aboveground biomass from very low-resolution drone images and their corresponding digital elevation maps. As expected, we found it harder to make the simple feedforward ConvNets converge on these more difficult datasets. Therefore, we had to equip our ConvNets with more recent techniques used in advanced architectures, like Inception and ResNet, for reasonable performance. Apart from the architectural engineering, our second contribution was to devise a simple superpixel based data augmentation strategy under the assumption of similarity of minute sub-regions with similar spatial statistics, which we call randomize minimal region swapping (RMRS). In fact, RMRS made it possible for us to train the ConvNets with only 48 low-resolution, drone-plot images. Therefore, training of ConvNets with both high- and low-resolution outdoor plant images and the RMRS data augmentation strategy constitute the body of Chapter 3.

- **Chapter 4:** This chapter is a conceptual interlude from the main line of development for bolstering simple one-look

ConvNet models for robustness. In Chapters 2 and 3, we employed SegNet [14] for semantic segmentation that has a computation-costly decoder following the encoder sub-network. In general, semantic segmentation architectures are configured in this encoder-decoder style with the assumption that the decoder uses necessary features extracted by the encoder to produce the desired output. This assumption is both highly intuitive from the perspective of convolution and provides good results. However, this rule can be violated if ConvNets are regarded as merely sparsely-connected graphs. In that case, it might be possible to generate the segmentation map directly at the backend of the encoder with depth-to-space reordering of the spatial elements incurring no computational cost. Chapter 4 exhibits quite satisfactory results for simpler problems like binary segmentation with this perfectly reasonable, but somewhat iconoclastic view. The possible significance of our initial result lies into the heart of overfitting problem for smaller datasets. The lack of additional weight layers after the encoder (or, may be only one for smoothing) would make it straightforward to finetune pretrained encoders directly without the need for highly customized and tricky training procedures for lower cardinality problems like leaf segmentation used in Chapter 2.

- **Chapter 5:** In the previous chapters, we mostly focused on training one-look ConvNets, similar to the standard image classification architectures, to regress the scalar count from the images. Although weak spatial information are available to almost all the counting datasets (e.g. a dot annotation on the center of each object in the image), the loss formulation for those models only takes the difference between the absolute counts into account. The possible ramification of this overly simple, scalar measure is the complete lack of minimal guidance to the model about the properties of the target objects to search for. Consequently, simple one-look models tend to fail on seemingly ambiguous cases, i.e. harder instances as well as object-like sub-regions in the background. To alleviate this problem, in Chapter 5, we devise a weak or approximate loss based on Class Activation Maps (CAM), a well-known visualization trick, without using additional annotations, and use it alongside the rudimentary L1 loss, that eventually improves both quantitative and qualitative performance of the one-look models.

- **Chapter 6:** This chapter addresses a common challenge in many object counting datasets. These datasets, esp. aerial image databases, often comprise small number of high-resolution, variable-shaped training samples. While training the counting models, we mostly care about the number of instances, not the number of images and so, the receptive field just needs to cover the resolution of the largest possible instance in the dataset. Therefore, it is possible to train with randomly cropped patches of the high-resolution images. However, on inference, if we aggregate the counts by tiling the patches, it causes severe overestimate and underestimate because many object instances will be cut-off at the edge of the patches. However, in many cases, the opposite nature of these two errors helps nullifying each other, resulting in a false impression of low error over the full-resolution image. We coined this event “random nullification" due to its uncontrolled nature considering the weaker ground truth annotation available. Through comprehensive experimentation, we show that the unnormalized GAP (which we call Global Sum Pooling) can avoid this error in a very simple and elegant manner.

Finally, the thesis ends with possible future directions alongside concluding remarks.

1.3 Terminologies

Some of the terminologies and mathematical forms used in the later chapters are listed below:

- **Ablation study:** In medical science, "ablation" refers to the removal of some body tissues (esp. problematic ones) by surgery. However, in deep learning literature, an ablation study is the comparative performance analysis after removing various components of the models. This kind of analysis is used to understand the effect of different critical components in the proposed architectures.

- **$\mathbf{L_q}$ loss:** This is also known as the *Minkowski* loss [18]. The formulation is given by the following equation:

$$L_q(y,t) = \int\int |y(x) - t|^q p(x,t) dx dt \tag{1.1}$$

  where, x, y, t are input, output, and target variables, respectively. In the machine learning literature, L_1 and L_2 instantiations of this general loss formulation are used in practice.

2. Leaf Counting with Deep Convolutional and Deconvolutional Networks

In this chapter, we investigate the problem of counting rosette leaves from an RGB image, an important task in plant phenotyping. We propose a data-driven approach for this task generalized over different plant species and imaging setups. To accomplish this task, we use state-of-the-art deep learning architectures: a deconvolutional network for initial segmentation and a convolutional network for leaf counting. Evaluation is performed on the leaf counting challenge dataset at CVPPP-2017. Despite the small number of training samples in this dataset, as compared to typical deep learning image sets, we obtain satisfactory performance on segmenting leaves from the background as a whole and counting the number of leaves using simple data augmentation strategies. Comparative analysis is provided against methods evaluated on the previous competition datasets. Our framework achieves mean and standard deviation of absolute count difference of 1.62 and 2.30 averaged over all five test datasets.

2.1 Introduction

Traditional plant phenotyping, which involves manual measurement of plant traits, is a slow, tedious and expensive task. In most cases, manual measurement techniques use sparse random sampling followed by the projection of those random measurements over the whole population which might incorporate measurement bias. Further, plant phenotyping has been identified as the current bottleneck in modern plant breeding and research programs [35].

Therefore, interest in image-based phenotyping techniques have expanded rapidly over the past 5 years. Automation of the estimation of these visual traits up to a satisfactory level of accuracy using suitable computer vision techniques can boost production speed and reduce costs since fewer field technicians would be required for manual measurement each year.

In this chapter, we work on estimating the number of leaves on a plant at the rosette stage, which is an indicator of plant health [68]. Our main objective is not only to develop a robust computer vision model, but also to generalize it so that the plant breeders can use this framework regardless of the plant species they are working on and of the quality of the image data they have acquired. Like one of the previous works [37], we also pose this problem as a nonlinear regression problem, where given the images, our framework approximates the count directly without segmenting individual leaf instances. This regression hypothesis is useful for a couple of reasons. First, although this nonlinear regression problem appears to be very high dimensional, it is usually more efficient than counting by identifying the individual leaf instances. Second, from the perspective of supervised machine learning, collecting ground-truth leaf counts is much simpler than

generating ground-truth segmented regions for each leaf in the color images. In section 2.4 of this chapter, we show that the performance of the systems developed under the regression hypothesis is comparable to the state-of-the-art counting by instance segmentation approaches. However, unlike [37], we develop each of the components of our complete model in such a way that it can directly learn from the data without the need for manual heuristics or explicit knowledge on the plant species or other environmental factors. According to the state-of-the-art computer vision and machine learning literature, the best way to develop a generalized model without such prior knowledge is to use deep learning and therefore we adopt this paradigm in our work.

Similar to [106], we train a deep convolutional neural network to count leaves by regression. However, the focus of our present work is to develop a single network that can generalize across different rosette datasets, rather than separate networks each built and tuned to maximize performance on an individual dataset. We also develop a deep convolutional-deconvolutional neural network for automatic whole plant segmentation and explore the effect of using a binary segmentation mask as an additional input channel to the leaf counting network in order to improve generalized performance. We evaluate our method as part of the Leaf Counting Challenge 2017 (LCC-2017) and report performance across the five subsets of the competition dataset. Through this work, we hope to inaugurate the research and development of a useful and generalized system for plant breeders to study leaf development in individual plants and eventually to study crop emergence in the field.

2.2 Related Work

We classify the recent literature performing leaf counting either directly or via instance segmentation into three categories, i.e. Leaf Segmentation Challenge in CVPPP-2014 (LSC-2014), Leaf Counting Challenge in CVPPP-2015 (LCC-2015), and others. Below we provide a brief account of the methods under each of these categories.

LSC-2014: In total, 4 methods evolve from this competition [89]. Although the training dataset for the competition included individual leaf instances indicated by different colors as the ground-truth, none of the 4 approaches use that ground truth to solve the instance segmentation problem. From that standpoint, they are all are eligible for the LCC-2017 competition also. The winner of this competition is IPK [74, 89]. This method utilizes $3D$ histogram of the Lab color space of the training images to model both plant regions and background and test pixels are inferred non-parametrically using direct interpolation on the training data. Then, leaf centers are extracted using mathematical morphology of the distance map of the segmented foreground. These centers along with the foreground segmentation are processed by heuristics-based graph algorithms to generate final instance segmentation map. Next, comes the unsupervised Nottingham approach [89], which segments the foreground using seeded region growing [7] over the superpixels [6] extracted from the Lab color map. For the subsets of the dataset containing non-overlapping images, empirical thresholds are used instead of the superpixel means as the initial seed. Like IPK [74], they compute the distance map over the foreground pixels. Then, superpixels with centroids nearest to the local maxima in the distance map are chosen as the initial seeds with the assumption that they represent leaf centers the best for watershed based instance segmentation [107]. The MSU approach is adopted from the literature on multiple leaf alignment and tracking [113–115] and primarily based

on template matching based on Chamfer Matching algorithm [15]. The authors use empirical threshold on the “a" plane of the Lab image to select foreground candidates on which template matching is performed. The main drawbacks of this approach are manual selection of both the threshold and the templates and exhaustive template matching with a large number of templates, i.e. 1080 templates for 2 subsets and 1920 for another. The last method submitted in LSC-2014 is Wageningen [89]. To segment the plant regions, the authors of this approach train a simple artificial neural network comprising one hidden layer of 10 units with six pixel-based features, i.e. red (R), green (G), blue (B), excessive green ($2G - R - B$), and variance and gradient magnitude of filtered green pixel values, and then post-process the network output using morphological operations with heuristically chosen parameters. After that, watershed transform [39] followed by empirical threshold based merging is performed to produce the instance segmentation result. A limitation of this method is the use of simple pixel features for foreground segmentation without using any contextual information in depth, resulting in the heavy usage of morphology to fine-tune the network output afterward.

LCC-2015: Only the winning method of LCC-2015 competition, General Leaf Counting (GLC) [37], is published in CVPPP-2015. To the best of our knowledge, this is the first approach posing the leaf counting problem as a nonlinear regression problem. The authors transform the original RGB image into a log-polar image [11] prior to further processing it to exploit the radial structure of the plants. Next, from the log-polar image, they extract patches based on the ground-truth foreground-background ratio in a sliding window fashion. These patch features are further vectorized with K-means [18] and triangle encoding [25]. Lastly, max-pooling over the patch features is performed to form the final feature vector for each image and a support vector regression network [28, 37] is trained for the prediction task. A limitation of this system is that the authors use ground-truth plant segmentations in both training and testing phases of the counting module. While approximate plant segmentations could be generated by other methods [67], the study used perfect segmentations and therefore it is not clear how robust their counting module is to noisy or imperfect segmentations that are typical of automatic segmentation procedures.

Others: All the methods proposed since LCC-2015, addressing either the direct counting problem or counting by instance segmentation are found to be based on deep learning, which is not surprising given the resurgence of this subfield of machine learning in recent years. In the recurrent instance segmentation (RIS) approach [84], the authors harness the power of sequential input processing of recurrent neural networks (RNN) [40] with the convolutional version of LSTM cells [45] to segment out one leaf instance at a time. Unlike the use of LSTM and RNN in natural language processing, the idea is to use convolutional LSTM instead of the original formulation to facilitate the training of the network by mitigating the computational complexity of fully connected layers as well as exploiting the semi-global statistical properties of images. To deal with the problem of possible ordering of individual instances in the image, the authors formulate the loss function based on the relaxed version of intersection over union (IoU) [53] and cross-entropy. The work done by Ren and Zemel [82] also use RNN similar to RIS [84]. However, their approach is primarily focused on extracting small patches each time to segment one instance using a similar idea of recurrent attention model [69] and then processing that small patch with LSTM [45] and a deconvolutional network [72] like architecture to segment a single instance. At the time of this writing, this work demonstrates the state-of-the-art performance for instance segmentation. Both this work and RIS use instance-level ground truth to train their networks and are therefore not

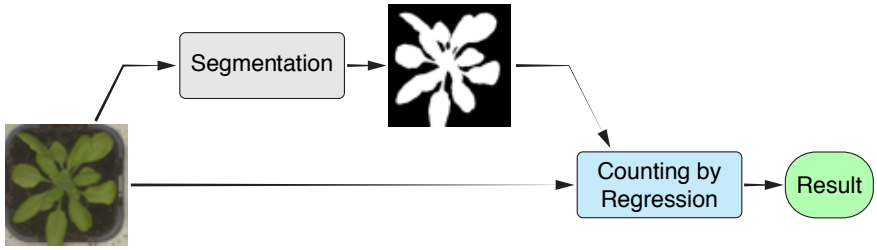

Figure 2.1: Block diagram of our approach.

directly comparable against ours. Nonetheless, we list their performance results in the *Experiments* section. Finally, the deep plant phenomics (DPP) approach [106] proposes a method addressing the problem of counting directly without both plant segmentation and instance segmentation. The authors customize their architectures as well as input dimensions to achieve state-of-the-art accuracy on different subsets of the LCC-2015 dataset. However, it is not known if the approach would generalize and if a single DPP network would provide consistent results across all datasets. Moreover, their training strategy relies on certain assumptions based on the nature of the images available in the LCC-2015 dataset; therefore, it is not clear how this approach would perform on the new types of images in the LCC-2017 competition. We will provide a detailed discussion about these issues while comparing our framework to DPP in section 2.4.

2.3 Our Approach

The approach presented in this section is developed to participate in the LCC competition [5] at CVPPP 2017. The high-level design of our framework follows a traditional computer vision workflow where the segmentation module is followed by the counting module (Figure 2.1). Within each module, we incorporate task-specific convolutional architectures, which are trained without explicit knowledge of the plant species to develop a generalized framework able to learn only from the data. The architectures used for segmentation and counting are trained separately, but not independently since the binary mask generated by the segmentation model is used to train the counting model in conjunction with the RGB channels. In the following two subsections, we will describe the architectures along with the rationale behind their design. Training methodologies and data augmentation strategies for these models are described in the experiments section.

Figure 2.2: Sample images from the training set of CVPPP-2017 dataset [5, 16, 68, 88]. Representative images are taken and scaled from 4 training directories $A1$, $A2$, $A3$, and $A4$, respectively.

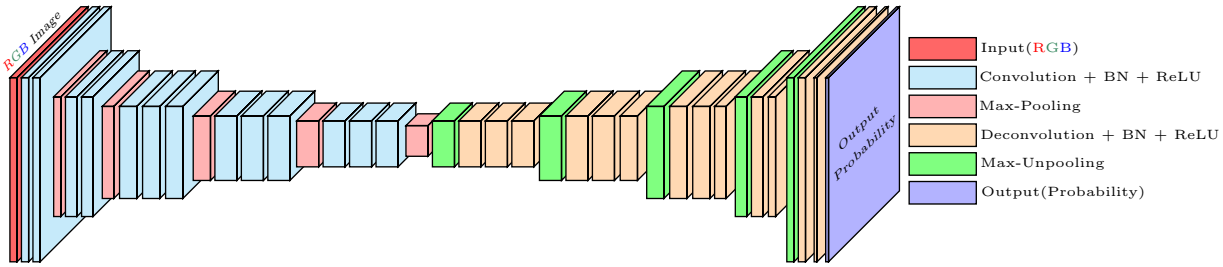

Figure 2.3: SegNet architecture [14] used for leaf segmentation. Each of the convolution and deconvolution layers is followed by batch normalization (BN) [48] and rectified linear unit (ReLU). All the pooling operations are 2×2 max-pooling with stride of 2. Similarly, the unpooling operations are 2×2 max-unpooling using the pooled indices taken from their corresponding max-pooling operations in the front-end of the network.

2.3.1 Segmentation

The segmentation problem we address is that of differentiating the plant or foreground pixels, from the background. This kind of problem is also known as semantic segmentation where the semantics of the objects are utilized to accomplish the task. In recent years, many papers [23, 65, 120] have been published addressing the solution for semantic segmentation from RGB images. Some of these architectures belong to the class of neural networks called *deconvolutional networks* [14, 72]. The main idea behind this kind of network is to construct a compact and informative set of feature maps or vectors from a set of input images, and then generate class-probability maps from the feature maps. Like other convolutional networks, construction of the feature set from the input data is done by a convolutional sub-network comprising multiple layers of convolution, pooling, and normalization operations. This convolutional sub-network is followed by a deconvolutional sub-network consisting of convolution-transpose, unpooling, and normalization operations to generate the desired probability maps. From the standpoint of semantic segmenatation, both height and width of the input and the output are the same. Hence, the deconvolutional part of the network is designed as a mirrored version of the convolutional part, except the input and the output layers, irrespective of the complexity of the problem and the dimensionality of the class-space.

Usually the design of a deconvolutional network contains fully connected (FC) layers in the middle to generate the feature vector from the pooled feature maps [72]. The FC layers are used to extract features in the global context for segmentation, and are therefore important if global context is necessary for the segmentation task. However, we propose that features in the semi-global context should be sufficient to segment the leaf regions from the background in color images, and therefore the FC layers could be omitted for our application. An advantage of eliminating the FC layers is that it considerably reduces the number of trainable parameters without sacrificing performance. For these reasons, we adopt the SegNet architecture [14], which omits FC layers and has shown promising results on SUN RGB-D [101] dataset comprising complicated indoor scenes and CamVid [20] video dataset of road scenes. The removal of FC layers in SegNet results in about 90% reduction of the number of trainable parameters as well as computational complexity. Figure 2.3 depicts the segmentation network we employ. The front-end convolutional sub-structure of the network is the VGG architecture [98] with batch normalization followed by each convolutional layer. In the convolutional front-end of

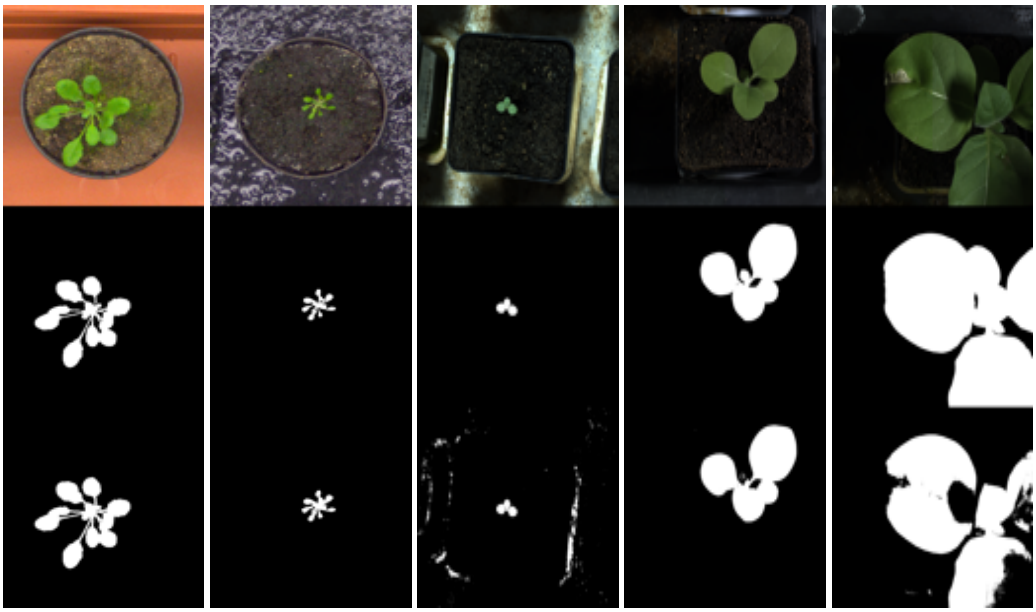

Figure 2.4: Sample images with corresponding binary segmentations: original RGB images (top row), corresponding ground truth segmentations (middle row), and our segmentation results generated by SegNet (bottom row).

SegNet, there are five 2×2 pooling operations with zero overlapping following multiple convolution and rectification layers each time. Hence, the convolved feature maps are compressed 32 times before starting the decompression via the deconvolutional back-end. We hypothesize that such level of compression or semi-global consideration is qualitatively sufficient to solve a comparatively easier problem of whole plant segmentation (Figure 2.2) as compared to other domains of semantic segmentation.

2.3.2 Counting

As shown in Figure 2.1, we use both the RGB image and the corresponding binary segmentation image to estimate the number of leaves after the segmentation is done. The rationale behind providing the counting module with the segmentation mask and the original RGB image instead of providing either the segmented region in the RGB image or the binary mask alone will be evident from Figure 2.4. Although the segmentation results generated by SegNet are sufficiently accurate for the counting phase for many images in the dataset, our network generates spurious segmentations for few of them. The poorly segmented images generally have lower average intensities and regions of leaves where the color and texture properties are washed out or blurred. We expect our network to do more or less accurate segmentation for these images by using semi-global contextual information, but we believe it fails due to the low number of available samples of that kind in the training dataset both in terms of absolute count and ratio of the samples of this particular kind to other kinds. The problem of this data scarcity is specific to the data-hungry approaches like deep learning, which requires a substantial number of training instances of a particular prototype to generate an accurate input-output mapping for that specific type.

Therefore, providing both the segmentation and the original image as input, we hope to influence the network to

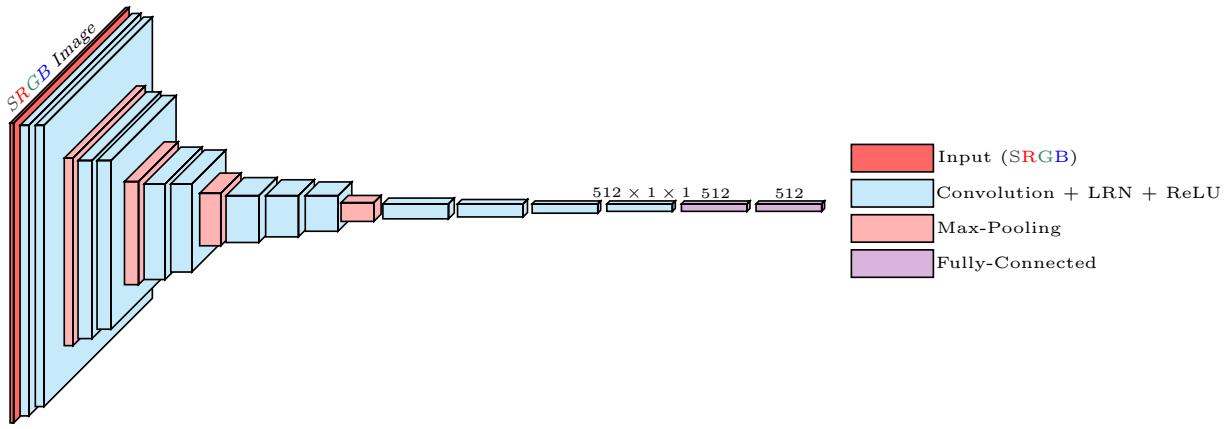

Figure 2.5: Counting architecture used for estimating the number of leaves from SRGB (Segmentation + RGB) channels. Each of the convolution blocks is a combination of convolution, local reponse normalization [55], and rectified linear unit (ReLU). All the pooling operations are 2×2 max-pooling with stride of 2.

recover the missed plant regions as well as reject the false detections from original image with the help of segmentation mask for counting. We call this four channel input as the SRGB (Segmentation + RGB) image. We also expect that providing the segmentation channel as input to the leaf counting network will help to suppress bias from features in the background of the training images, such as the soil, moss, pot/tray color, which will vary between datasets.

The design of our leaf counting by regression network takes inspiration from the VGG architecture [98], which reinforces the idea of deeper architectures with a long list of convolutional and rectification layers stacked one after another with several pooling layers in between and then the classification layer follows a couple of fully connected layers. Usually, this kind of convolutional networks use suitable amount of padding to maintain fixed height and width of the feature maps. Padding the input maps serves well when the network is trained with large-scale datasets containing samples in the order of millions. However, in our case, we have a small dataset of several hundred images to train, which is very difficult to augment beyond several thousand images. Hence, to retain the power of deeper architecture and to train the parameters without significant overfitting at the same time, we reduce the number of parameters effectively by using convolution without padding throughout the network. Moreover, we choose the filter size of the convolutional layers in such a way that before proceeding through the fully connected layers, the feature map turns into a vector. Thus, with zero padding and careful choice of filter size, we are able to reduce the number of parameters from $49M$ to $30M$. Implementation details for both segmentation and regression networks are provided in the following section.

2.4 Experiments

In this section, we provide a detailed account of our experimental setup. First, we describe the dataset used for evaluation. Next, the training strategies for both networks are specified. Finally, the performance of our framework is analyzed and compared against state-of-the-art literature from both quantitative and qualitative standpoints.

2.4.1 Dataset

The dataset we use to evaluate our framework is provided to the teams registered for the Leaf Counting Challenge (LCC-2017). The objective of this challenge is to come up with the solutions able to count the number of leaves from plant images directly via learning algorithms without detecting individual leaf instances. All the RGB images in the dataset belong to either Tobacco or Arabidopsis plants. For the LCC competition, each RGB image is accompanied by a binary segmentation mask with 1 and 0 indicating plant and background pixels, respectively, and a center binary image with leaves centers denoted by single pixels.

The training dataset is organized into 4 directories, namely $A1$, $A2$, $A3$, and $A4$. Directories $A1$ and $A2$ contain Arabidopsis images taken from growth chamber experiments with larger but different field of views covering many plants and then cropped to a single plant. Directory $A3$ enlists the Tobacco images with the field of view chosen to encompass a single plant. $A4$ is a subset of another public Arabidopsis dataset [16] collected using a time-lapse camera. In total, there are 27 Tobacco images in $A3$, and 783 Arabidopsis images in the rest of the directories. The organizers denote these directories along with the images as “SPLIT" images since they are split into separate folders according to the origin. In addition, all these directories contain CSV files including ground truth leaf counts under the same nomenclature.

The “SPLIT" directory structure for the testing set is the same as training, except that it includes an extra directory denoted by $A5$, enlisting images from different sources of origin altogether with the objective to emulate a leaf counting task in the wild. Hence, the organizers represent $A5$ images under the nomenclature “WILD".

2.4.2 Training and Implementation

SegNet training: Unlike training in the original SegNet paper [14], we trained our model from scratch without using any pretrained weights for initialization. Also in SegNet, the authors used different learning rates for different modules, whereas a fixed learning rate was used for all the layers in our training.

We used an input and output image size of 224×224 pixels in SegNet, whereas the original image size was approximately 500×500 and 2000×2500. While training deeper networks, the obvious advantage of using smaller input-output size than the original ones is data augmentation up to a considerable amount. We augmented the data and train the network in 3 stages. First, for each image, we extracted the union of top 20 object proposals [123], flipped top-bottom and left-right, rotated them with an angular step size of 4 degree, cropped the largest square from the center position to avoid dark regions due to rotation, and created a couple of Gaussian blurred version and corresponding sharpened images. In this way, we generated about $0.8M$ augmented samples from 810 original images and trained the network for 5 epochs with randomly cropped 224×224 subsamples. Second, we took the proposal images and their flipped versions and generated nearly $0.3M$ subsamples of size 224×224 deterministically with a fixed stride and train the network for another 8 epochs. Finally, we generated another $0.19M$ samples in a similar manner as in the second step, but this time from the original images instead of the proposals. Then, we fine-tuned the network with these $0.19M$ samples for 37 epochs. In all stages, SGD-momentum was used as the optimizer with initial learning rate, momentum

and weight decay of 0.01, 0.9, and 0.0001, respectively and these parameters were changed later based on the training statistics. Spatial cross-entropy was used as the error criterion. The ratios of foreground to background weights in the cross-entropy calculation for the first stage was 2.0 and 1.2 for the later steps. In the test phase, we took dense 224×224 samples deterministically with fixed stride from each of the test images and classified each pixel based on the aggregate probability over the samples. We initialized the convolutional weights with Xavier [38] prior to the start of training.

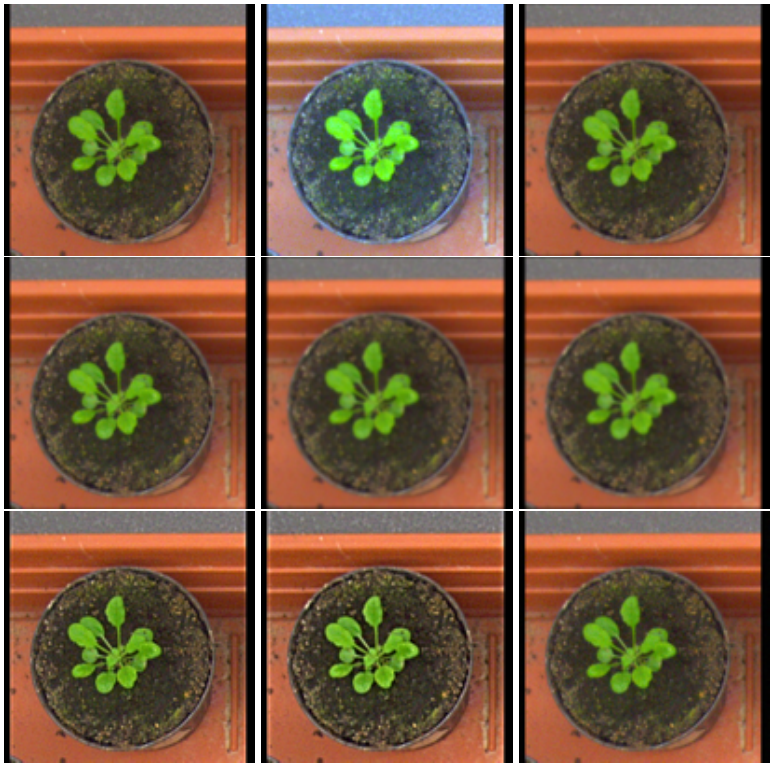

Figure 2.6: Augmentation samples for training the counting network.

Table 2.1: Head-to-head comparison against LCC-2015 winner. Note that *All* refers to A1-A3 for GLC and A1-A5 for Ours.

Directories	CountDiff		AbsCountDiff		PercentAgreement [%]		MSE	
	GLC [37]	Ours	GLC [37]	Ours	GLC [37]	Ours	GLC [37]	Ours
A1	-0.79(1.54)	-0.33(1.38)	1.27(1.15)	1.00(1.00)	27.3	30.3	2.91	1.97
A2	-2.44(2.88)	-0.22(1.86)	2.44(2.88)	1.56(0.88)	44.4	11.1	13.33	3.11
A3	-0.04(1.93)	2.71(4.58)	1.36(1.37)	3.46(4.04)	19.6	7.1	3.68	28.00
A4	-	0.23(1.44)	-	1.08(0.97)	-	29.2	-	2.11
A5	-	0.80(2.77)	-	1.66(2.36)	-	23.8	-	8.28
All	-0.51(2.02)	0.73(2.72)	1.43(1.51)	1.62(2.30)	24.5	24.0	4.31	7.90

Count network training: Training of the counting network is fairly straightforward compared to SegNet. In this phase, we used all the images as a whole without prior cropping or sampling operation for data augmentation to ensure that the ground truth leaf counts were valid for all augmented images. Also, while designing the network architecture,

Table 2.2: Comparison against state-of-the-art literature. Note that *All* refers to A1-A3 for previous work and A1-A5 for Ours.

Methods	CountDiff						AbsCountDiff					
	A1	A2	A3	A4	A5	*All*	A1	A2	A3	A4	A5	*All*
IPK [74]	-1.8(1.8)	-1.0(1.5)	-2.0(3.2)	-	-	-1.9(2.7)	2.2(1.3)	1.2(1.3)	2.8(2.5)	-	-	2.4(2.1)
Nottingham [89]	-3.5(2.4)	-1.9(1.7)	-1.9(2.9)	-	-	-2.4(2.8)	3.8(1.9)	1.9(1.7)	2.5(2.4)	-	-	2.9(2.3)
MSU [89]	-2.5(1.5)	-2.0(1.5)	-2.3(1.9)	-	-	-2.3(1.8)	2.5(1.5)	2.0(1.5)	2.3(1.9)	-	-	2.4(1.7)
Wageningen [89]	1.3(2.4)	-0.2(0.7)	1.8(5.5)	-	-	1.5(4.4)	2.2(1.6)	0.4(0.5)	3.0(4.9)	-	-	2.5(3.9)
GLC [37]	-0.79(1.54)	-2.44(2.88)	-0.04(1.93)	-	-	-0.51(2.02)	1.27(1.15)	2.44(2.88)	1.36(1.37)	-	-	1.43(1.51)
DPP [106]	-	-	-	-	-	-	0.41(0.44)	0.61(0.47)	0.61(0.54)	-	-	-
RIS+CRF [84]	-	-	-	-	-	0.2(1.4)	-	-	-	-	-	1.1(0.9)
EERA [82]	-	-	-	-	-	-	-	-	-	-	-	0.8(1.0)
Ours	-0.33(1.38)	-0.22(1.86)	2.71(4.58)	0.23(1.44)	0.80(2.77)	0.73(2.72)	1.00(1.00)	1.56(0.88)	3.46(4.04)	1.08(0.97)	1.66(2.36)	1.62(2.30)

Table 2.3: Possible interpretation of the performance measures.

Measures	Possible Interpretation
CountDiff ↓	The model is less biased towards overestimate or underestimate.
AbsCountDiff ↓	Average performance is better.
PercentAgreement ↑	Number of accurate predictions is higher.
CountDiff ↓, AbsCountDiff ↓	Less bias with better performance. Desirable properties of an ideal nonlinear regression model.
CountDiff ↓, AbsCountDiff ↑	High positive and negative errors cancel out. Model behaviour tends to be linear than usual.
PercentAgreement ↓, AbsCountDiff ↓	Although many predictions are not exactly accurate, all of the predictions are close to the original; therefore, model performance is uniform over the samples.
PercentAgreement ↑, AbsCountDiff ↑	Although many predictions are exact, wrong predictions are far from the original; therefore, model performance is not uniform over the samples.

we experimented with adaptive operations to deal with variable sized images, but they did not seem to work better than resizing the images to a fixed size. Moreover, we had to be cautious in the choice of the size for resizing operation so that for bigger images with resolution like 2000×2500, properties of the small leaf regions did not deteriorate much. Considering this fact, we chose the modified image size to be 448×448 preserving the aspect ratio. Thus, the largest dimension was taken to be 448 and the smaller one was padded with zeros afterward.

After the resize operation was performed, each of the images was augmented 8 times using intensity saturation, Gaussian blurring and sharpening, and additive Gaussian noise (Figure 2.6). Each image was also flipped top-bottom and left-right and rotated 180° along with similar augmentations. Thus, we generated 36 slightly different samples with the same ground truth from each original image, resulting in 29160 training instances for the regression network.

After the data generation was done, the counting or regression network was trained for 40 epochs using Adam [51] with fixed learning rate and weight decay both set to 0.0001. Smooth-L_1 criterion was used as the loss function instead of simple L_1 criterion to prevent gradient explosions as described in [36]. At first, we started training the model with normalized FC layers of size 1024. However, based upon the training statistics and to reduce the risk of overfitting, we changed the size of FC layers to 512 and retrained the model with the already trained convolutional weights. Finally, the

model trained until epoch 35 was used to generate the prediction for final submission.

Implementation: We used Torch [27] as the deep learning framework for both models. All the convolutional filters of the segmentation network were of size 3×3. For regression architecture, 9×9 convolution was performed until the second max-pooling operation and 5×5 afterward. We used the convolutional stride of 1 throughout both networks. All the pooling operations were 2×2 max-pooling with stride of 2. The dimension of all fully connected layers in the regression network was 512. Training was performed on a single NVIDIA Quadro P6000 Dell workstation. On this machine, training of SegNet took about $6 - 7$ days, whereas the regression network was trained within a couple of days.

2.4.3 Evaluation

Evaluation of our complete framework was accomplished in three stages. First, we assessed the segmentation network in terms of the precision and recall (equation 2.1) of the plant pixels. Next, we performed a head-to-head comparison against the winner of the previous LCC competition. Finally, we compared our results to the state-of-the-art approaches. We also performed an ablation study by training our counting network with and without the segmentation channel as input, in order to cast some light on the issue regarding the need for foreground segmentation.

Foreground segmentation: Even though the accuracy of binary segmentation is not a criterion for evaluation in the LCC competition [5], we provide precision and recall (equation 2.1) of our segmentation model in Table 3.1 to justify our assumption on the sufficiency of semi-global context for leaf segmentation. It is evident from Table 3.1 that the segmentation results generated by SegNet using semi-global information are good enough to be used for the regression network. Performance of the segmentation network is comparatively lower for directory $A3$ (Table 3.1, red text) since there are only 27 Tobacco images in the $A3$ training set as compared to 783 Arabidopsis images in the rest of the directories.

$$\begin{cases} \text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \\ \text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \end{cases} \tag{2.1}$$

Table 2.4: Binary segmentation results.

Directory	A1	A2	A3	A4	A5
Precision	0.98	0.94	0.80	0.96	0.92
Recall	0.99	0.99	0.94	0.98	0.97

Comparison against the previous winner: Next, we provide comparisons in both Table 2.1 and 2.2. Table 2.2 provides comparisons against all the recent literature, whereas Table 2.1 provides a head-to-head comparison against the LCC-2015 winner, which is more detailed due to the availability of the performance metrics for [37]. In Table 2.1, "CountDiff" refers to the mean and standard deviation (shown in parentheses) of the difference in count averaged over images. "AbsCountDiff" is the absolute of "CountDiff". The term "PercentAgreement" indicates the percentage of exact matches between the actual prediction and ground truth measurement for counts. "MSE" is the abbreviation for

mean-squared error.

From Table 2.1, it is evident that we achieve lower CountDiff and AbsCountDiff for directories $A1$ and $A2$. Lower CountDiff means that our model is less biased towards underestimation or overestimation than GLC [37], whereas lower AbsCountDiff can be interpreted as the indicator of better average performance of the system. However, our framework performs poorly on directory $A3$ (Table 2.1, red text). The reason behind the failure is pretty straightforward. Note that, in the training set, there are in total 783 Arabidopsis images in $A1$, $A2$, and $A4$. On the other hand, there are only 27 Tobacco images in $A3$, which is scarce for the types of deep architectures we are using that contain millions of parameters. Hence, our regression network fails to model the distribution for leaf counting over the Tobacco images. This inadequacy is also reflected in the AbsCountDiff measure for the test directory $A5$, which is a mixture of Arabidopsis and Tobacco images altogether.

For directory $A2$, although our CountDiff and AbsCountDiff are better than those of GLC, PercentAgreement of GLC is much better than ours. Apparently, it might seem to be a pitfall of our system. However, the combination of lower AbsCountDiff and lower PercentAgreement means that even though the number of exact predictions is low, all the predictions are pretty close to the original and the overall performance of the system is more or less uniform over the test images. On the contrary, comparatively higher values of AbsCountDiff and PercentAgreement, which belong to GLC for directory $A2$, refers to the situation where model performance is not uniform over the samples. In other words, predictions may be accurate for easier samples with no leaf overlap or moderate-sized leaves or both, but deteriorate for harder cases with smaller or overlapping leaves. In that sense, our generalized framework is capable of modeling and inferring leaf shapes under deformation and partial occlusion better than GLC given a few hundred images for a particular species. To facilitate this kind of comparative evaluation of our method by the readers, we enlist a set of combinations of the measures along with their possible interpretations in Table 2.3. Also, note that our average measurement (directory "*All*") is over 501 test images from 5 directories (A1-A5), whereas the average for GLC is taken over 98 test images from 3 directories (A1-A3).

General comparison: Table 2.2 shows that our method performs well as compared to all the LSC-2014 [74, 89] and LCC-2015 [37], except for the failure on directory $A3$ due to inadequate number of samples. Both RIS+CRF [84] and EERA [82] use instance-level ground truth. Hence, they are eligible for the segmentation competition (LSC), but not the counting competition (LCC). Nonetheless, we put them in the list to demonstrate our comparability to these state-of-the-art methods developed with instance segmentations that are more expensive in terms of training complexity/time and ground truth data requirements. DPP [106] is the only method close to ours in the style of approach, except that they use three shallow regression networks, each one highly customized over a single directory. Moreover, DPP uses random cropping from $10\% - 25\%$ for the purpose of data augmentation while training. This could result in mislabeled images if leaves are cropped out of certain images. The new rosette images in the LCC-2017 dataset include larger rosettes that cover more of the image frame (and extend outside the frame in certain cases, see rightmost image in Figure 2.2); therefore it is not clear how DPP would perform on the larger and more varied test images in the new competition dataset.

Ablation study: To justify the inclusion of a segmentation network within our framework, we performed an ablation

study by training our regression network using only RGB images as the input without foreground segmentation. We found slower convergence than that of using the segmentation images as input. However, counting results using only RGB images were comparable, which supports the approach proposed by DPP of using a regression network directly on RGB images and that the network learns relevant features directly without a priori segmentation. Nonetheless, we do expect that providing foreground segmentation as an additional input channel helps to push the regression architecture to train on localized features within the plant region in the image. This might help to suppress background features that could limit the generalizability of the counting model if provided images of rosettes grown in different backgrounds, e.g. in different pots, trays, or growth tables. The issue of localization of features in these types of regression networks requires additional attention as future work.

2.5 Conclusion and Future Work

In this chapter, as a participant of the LCC-2017 competition, we provide a complete and generalized data-driven framework for leaf counting from RGB images directly without instance segmentation. We demonstrate that given a moderate amount of data on any species, our architectures are able to learn to estimate the number of leaves without prior knowledge on that particular species or surroundings of the plant. From the perspective of informed search strategies, we do plant segmentation prior to counting with the assumption that the additional foreground segmentation channel guides the regression model to extract necessary features only from the plant region and thus trains the model correctly. However, based upon other recent works and ours, the need for segmentation prior to counting by the deep networks is still an open question. As future work, we plan to investigate this issue in more detail, with the goal of achieving equivalent performance to that of instance segmentation architectures with much simpler and easier to train non-recurrent networks such as reported in the present study.

3. DeepWheat: Estimating Phenotypic Traits from Crop Images with Deep Learning

In this chapter, we investigate estimating emergence and biomass traits from color images and elevation maps of wheat field plots. We employ a state-of-the-art deconvolutional network for segmentation and convolutional architectures, with residual and Inception-like layers, to estimate traits via high dimensional nonlinear regression. Evaluation was performed on two different species of wheat, grown in field plots for an experimental plant breeding study. Our framework achieves satisfactory performance with mean and standard deviation of absolute difference of 1.05 and 1.40 counts for emergence and 1.45 and 2.05 for biomass estimation. Our results for counting wheat plants from field images are better than the accuracy reported for the similar, but arguably less difficult, task of counting leaves from indoor images of rosette plants. Our results for biomass estimation, even with a very small dataset, improve upon all previously proposed approaches in the literature.

3.1 Introduction

Measuring the phenotypic traits of crops, which are the differences in plant characteristics caused by the interaction of the plant's genetics and the environment, is important in plant breeding research as it allows the breeders to select crop varieties with desirable physical characteristics, such as high yield, resistance to stress, and ability to be easily harvested. Traditionally, phenotypic measurements are made manually in the field, which is both labor intensive and potentially inaccurate due to substantial sub-sampling involved. To overcome these drawbacks, image-based automated phenotypic traits estimation is emerging as an important area of applied computer vision research with the goal of capturing more accurate information at a large scale for better crop production.

In many crops, including wheat, emergence (the density of plants within the field) and biomass (the total mass of each plant) are important phenotypes. Emergence is important because a vigorous and uniform crop stand is needed to compete for moisture, nutrients, and sunlight. Plants that emerge late will have a lower yield than the early emerging ones due to the increase in competition for sunlight and essential nutrients [58]. Determining biomass in different crop varieties is important because it is correlated with yield [102] and photosynthetic activity, and is an indicator of overall plant health [30]. These phenotypes are labour intensive and destructive to measure manually: emergence typically requires physically touching plants in the field to determine which leaves belong to which plant, and biomass measurements are made by cutting out plants from the field and measuring their mass. Furthermore, these phenotypes are traditionally measured on only a small sub-sample of the experimental plot area, which can result in sampling error.

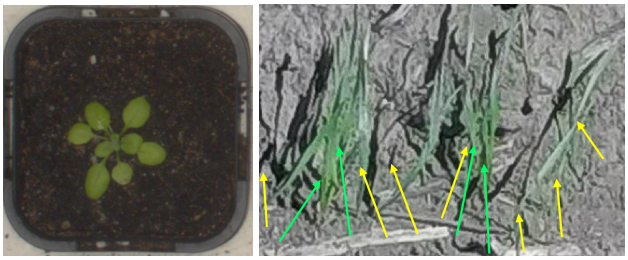

Figure 3.1: Eleven leaves in an image from the standard leaf counting dataset [16] (left) and eleven wheat plants in an outdoor image used for emergence counting in this chapter. Counting plants from the right image is more challenging to due to variable number of leaves per plant and occlusion.

The combination of high importance and high measurement difficulty makes these phenotypes good candidates for image-based phenotyping in any crop breeding programs.

Counting plants is related to the well-studied problem of counting leaves from plant images [9, 37], but much more challenging. Wheat seeds are planted in close proximity, therefore, the plants grown from these seeds are highly occluded by each other in the image. To illustrate the level of difficulty, Figure 3.1 shows a sample image from the standard leaf counting dataset [16] and another image from the dataset we are using for wheat emergence counting. Both images have the same label: 11 leaves in the left image, and 11 wheat plants in the right image. In the left image, the number of leaves is unambiguous despite a few small leaves in the center, which is not the case for plant count in the right image. According to the plant science experts who generated the ground truth counts and who have experience counting plants in the field, while counting from the images, they looked at the stems as close to the ground as possible. When a stem seemed unreasonably thick, they presumed that there were more plants behind the visible ones. Plant bases indicated by the yellow arrows in the figure are easy to count. However, in regions denoted by the green arrows, it may look like there is one plant, based on the thickness of the plants, amount of leaves, and age of plants, the count of plants was estimated by the raters as more than one. Hence, both intuition and experience play a role in accurate emergence counting, making it a difficult image analysis task.

In this chapter, we propose completely data-driven frameworks for emergence counting and biomass estimation. We develop generalized architectures for phenotypic traits estimation blending the concepts of learning sparse structure via dense, multiscale representations [103] and residual or shortcut connections [44]. We train our models from scratch to keep our phenotypic estimation tasks independent of the other large-scale machine learning tasks pursued with very large models. For this reason, to efficiently train the data-hungry deep models with a few training samples, we also propose a novel data augmentation strategy based on randomized minimal region swapping of the superpixels in an image, which can be used to augment low to medium resolution images. Also, we examine the quality of learning of the emergence counting architecture qualitatively by visualizing salient regions using the class activation mapping (CAM) [122] approach. We find that the learned network features focus on image regions that are responsible for counting, notably the base of each leaf-cluster, and the dense regions of leaves, according to the plant breeding experts who provided the ground truth counts.

To the best of our knowledge, this is the first work on image-based phenotypic trait estimation of crops with deep

learning. The name *DeepWheat* refers to our overall system because of the first use of deep learning in this domain and since we have used the image dataset of two species of wheat for the evaluation. Although we evaluate our approach on wheat, our design allows the frameworks to be generalized to other types of crops with minimal additional manual intervention.

3.2 Related Work

Despite the significance of emergence and biomass in crop breeding, little computer vision research has been done on the automated estimation of these traits from images. Leaf counting has been studied in more detail due to a standardized dataset of rosette plants and previous computer vision competitions [5]. Recent approaches to leaf counting have employed convolutional neural networks to count by regression [9]. We adopt a similar approach in this study to evaluate if it extends to much more difficult phenotyping tasks such as plant and biomass counting from field images.

A few studies have looked at plant density estimation in maize [95–97] and wheat [49, 64] from RGB images. All of these previous methods employ a traditional image processing pipeline that requires hand-tuned parameters tailored to the specific crop of interest. In the wheat studies, the plant counting algorithm depends on the accurate segmentation of leaves, followed by extracting regional properties of the leaves as features, and then training a simple artificial neural network (ANN) [64] or a support vector machine (SVM) [49]. In both papers, the initial segmentation of the plant foreground from the soil background is accomplished with simple naive approaches: Otsu thresholding [39] on the “b" channel of Lab image or a predefined RGB transformation channel $(2G - 2B - 2.4R)$. However, simple threshold-based segmentations are not robust to variable illumination in different field environments. Indeed, these segmentation approaches are found to give very poor results for the images used in our study and are therefore not useful benchmarks for comparison.

A number of previous studies have attempted to estimate biomass, but most have done so from field-based measurements and are therefore not applicable to image datasets. A few studies have used aerial images as a basis for biomass estimation. In [90], naive linear regression models are fitted on plant height and plant coverage in aerial images. In [80], different linear and nonlinear combinations of height measured with an ultrasonic sensor, leaf area index measured with a plant canopy sensor, and vegetation indices from canopy reflectance obtained using a portable spectrometer are used as the predictors and biomass is used as the response of the multiple linear regression model. The product of leaf area index and dry matter content per leaf area is regarded as the estimation of above-ground biomass (AGB) in [79]. The authors also provide a comparison against the models developed using exponential regression, partial least square regression and simple artificial neural networks. In [57], AGB was estimated from height information obtained from the Digital Terrain Model (DTM) derived from LiDAR data. For each plot, simple statistical measures of height, such as mean, quadratic mean, standard deviation, skewness, kurtosis, and percentile of height along with height bins at fixed intervals, are used as the predictors for regression modeling. A similar approach is taken in [56] with additional vegetation indices extracted from hyperspectral data. In terms of the list of predictor variables, the approach in [17] can be considered an extended version of the other two [56, 57] with height information plus the vegetation indices based on

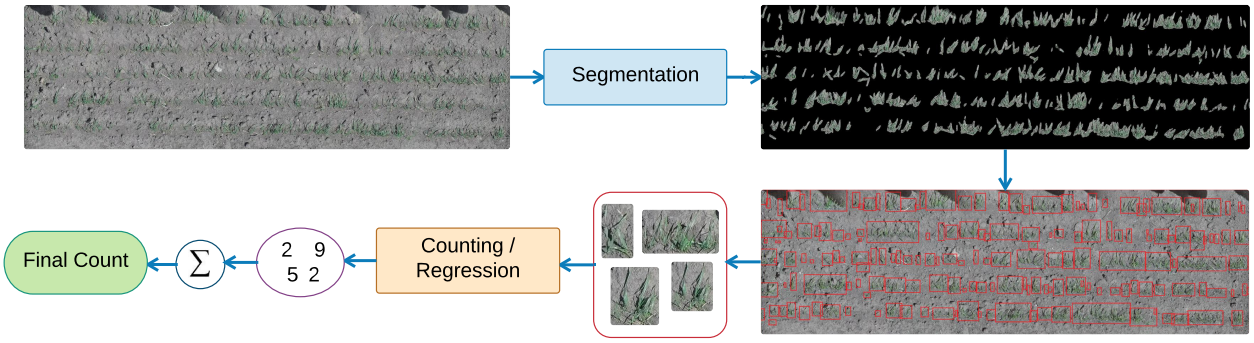

Figure 3.2: Workflow for emergence counting: 1) loosely segment the plant regions from RGB plot images with the segmentation module, 2) extract small patches containing plants via connected component analysis, 3) use counting module for individual counts on each patch, 4) sum all the patches to get the overall emergence count for a single plot.

both hyperspectral and unmanned aerial vehicle (UAV) images.

3.3 Our Approach

In this section, we describe the design of both emergence count and biomass estimation frameworks in detail. Although both traits are estimated by convolutional networks performing regression, the architectures and overall workflows are different.

3.3.1 Emergence Counting

Figure 3.2 depicts the overall computational procedure for counting crop emergence. First, we loosely segment the plant regions from the RGB plot images through the segmentation module described later. Next, we extract all the segmented patches from the whole image, as indicated by the red rectangles in Figure 3.2 and input each patch image to the counting module to get the individual emergence counts for each patch. Finally, we sum up all the predicted counts for a single plot image to get the overall prediction for emergence count for that particular plot. In this framework, both the segmentation and the counting modules comprise deep architectures which we describe below.

Segmentation

Our motivation for segmenting plot images into smaller patches is twofold. First, due to the very high resolution of plot images ($\sim 2500 \times 7500$), it is not computationally feasible to do the emergence counting task on the whole image at once. Instead, either sequential or parallel counting over disjoint plant regions is required. Second, data-driven approaches, like deep learning, require many training samples, whereas we have only a few high-resolution plot images available for that purpose. Therefore, we generate non-overlapping patches of segmented plant regions to provide us with more than a hundred subsamples from each plot image for further training of the counting model.

From the design perspective, we relax the output of the segmentation module from exact segmentation to a soft or relaxed segmentation for several reasons. First, generating the exact ground-truth manually for images like the ones

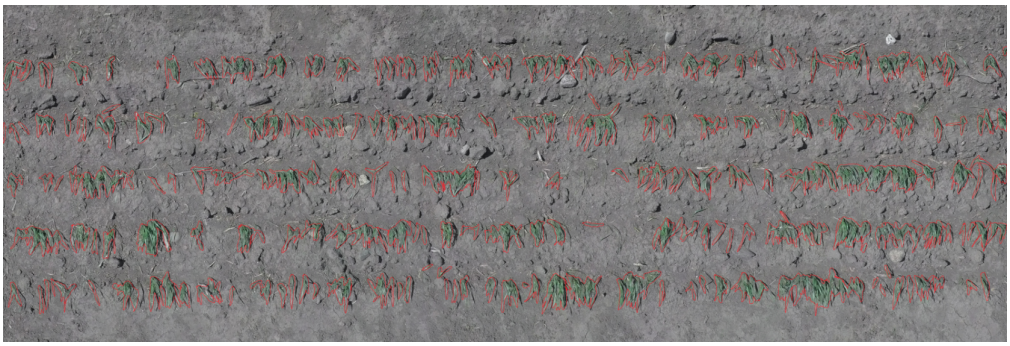

Figure 3.3: Manual ground-truth generated for relaxed segmentation of plants showing manually drawn contours around plant regions (red). Later, contours are filled with simple morphological hole-filling to create the binary segmentation mask.

shown in Figure 3.3 is a more tedious and time-consuming process than defining loose or relaxed contours around plants. Moreover, for deep networks, learning to count from the subsamples with exact vs. loose segmentations is similar since the background is uniform and so, it is unlikely that the model would pick up distinctive features from the background region. This claim is also validated by CAM [122] visualizations of the network in the *Experiments* section that show saliency in foreground regions. In addition, the wheat leaves are thin and partly occluded; therefore, going for precise segmentation could result in missing very thin or hard-to-detect regions of the plants which could deteriorate the counting performance since the model responsible for counting would assume the segregated leaves as different instances rather than a single one.

To perform soft segmentation with deep learning, we use the SegNet architecture [9, 14] rather than deconvolutional networks containing fully connected (FC) layers [72] with a many more training parameters. This is because the problem we are dealing with is easier than the exact segmentation and much simpler than general multi-class semantic segmentation both in terms of the cardinality of the output categories and the nature of the domain since the diversity of the pixel intensities in a single plot image is highly restrained compared to that of natural images. Furthermore, our concern is not to get an overall-high precision segmentation mask, rather we are concerned with not missing plant regions in the image for the counting model afterward.

Counting by Regression

In this chapter, we focus on different species of the crop wheat, which except the very late season, resembles mostly to grass crops. The leaves of such plants are the most deformable among all kinds of plants and crops, and so, a set of wheat plants in an image might appear in a combinatorially large number of variations. Thus, to successfully count the number of plants in the image, the deep model must be able to deal with such combinatorial number of deformations and resulting occlusions as much as possible.

As argued in the NIN paper [62], a simple stack of convolutional layers with an over-complete set of filters followed by nonlinearity and pooling serve well when the underlying concepts to be learned via abstract representation are linearly separable. However, for highly nonlinear latent concepts, replacing plain convolutional blocks with small networks

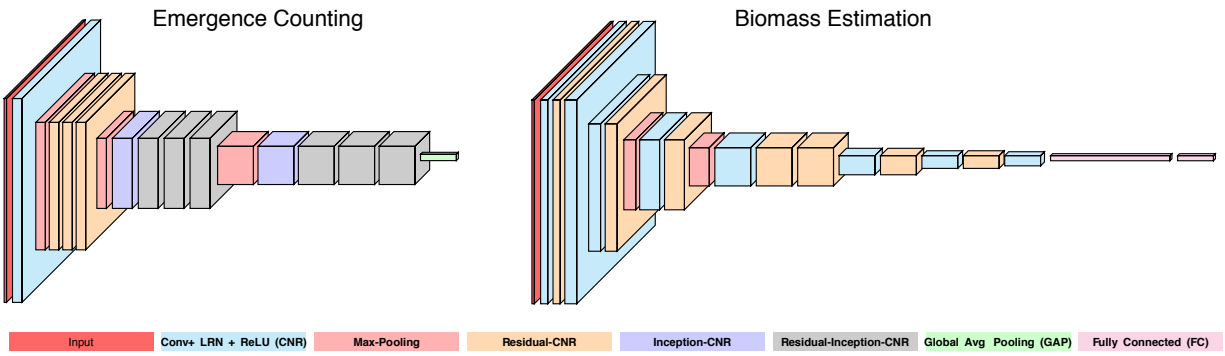

Figure 3.4: Emergence and biomass estimation architectures. We use 7×7 receptive fields in the initial CNR block with unit stride. The number of filters after each max-pooling operation is doubled, except the first one for emergence counting. *residual*-CNR is a simplified version of the residual block described in [44], where we keep the number of receptive fields constant inside the block. We use a simplified "Inception" module [104], where the number of input and output receptive fields are the same. Inside our Inception block, we employ half of the size of filters for 3×3 convolution, a quarter of the input size for the equivalent 5×5 convolution, and half of the rest for pooling and unit convolution each. For the emergence network, to visualize the representations learned by our model, we use global average pooling (GAP) [62].

inside the basic architecture is already proved to be useful in several large-scale image classification tasks [103, 104]. Hence, we take inspiration from these works, where the representation in each layer is approximated from the dense multi-scale feature responses learned in the previous layer. Also, we incorporate the concept of residual learning [44] in our architecture, which we experimentally found to be useful for faster training in case of stacked-convolutional architecture for our task.

Therefore, in the design of our network as depicted in Figure 3.4, four different convolutional blocks are used. Our initial convolutional block (CNR) is a simple convolution operation followed by local response normalization and rectified nonlinearity [55]. Next, we use a simplified residual version of the original residual block described in [44], in the sense that the number of feature maps is constant throughout the block from input to output. Also, for deeper layers, where the number of receptive fields is comparatively higher, we incorporate the "Inception" version of CNR followed by the *residual*-Inception version. All these modules are crafted to have the same input-output capacity. Finally, for the ease of visualization of the salient regions detected by our model, we simply use the global average pooling (GAP) [62] layer. We experimented with different setups of fully connected layers instead of GAP and got slightly improved performance. However, we prefer visualization over those minor improvements to encourage further research based on visualization. Lastly, we have not used any pre-trained model because unlike classification problems, the capacity of the final layer does not scale up with the complexity of the counting task. In addition, opening up the full network for finetuning might result in significant overfitting due to comparatively smaller datasets.

3.3.2 Biomass Estimation

For biomass estimation, we have both 5 channel orthomosaics (Blue, Green, Red, Near-infrared, red-Edge) and digital elevation maps (DEM). Sample RGB images are shown in Figure 3.5. The pixel values of the DEM files indicate the

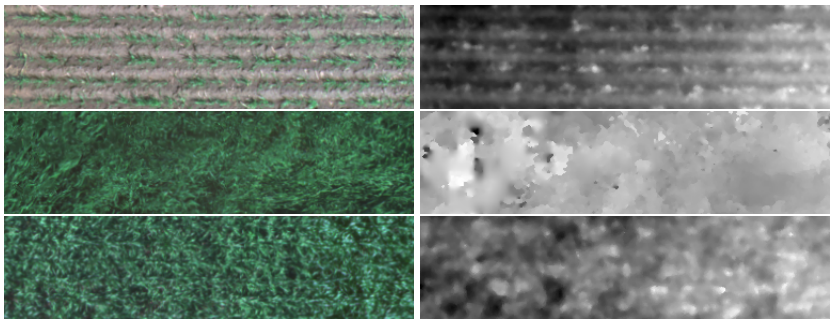

Figure 3.5: Sample RGB plot images (left) with corresponding DEMs (right) showing wheat plants from emergence as individual plants (top) to full crop canopy (middle) and during the reproductive stage (bottom). DEM values (height) converted to grayscale for visualization.

elevation of plants from the ground. Note that, the RGB images of the plots available for emergence counts in the previous section and biomass estimation here are from different sources. The plot images for biomass estimation are lower resolution ($\sim 120 \times 480$) than those used for emergence counting (see Section 3.4.1).

above-ground biomass refers to the weight of all plant material above the ground. We expect that there is a relationship between biomass and height or elevation values of the DEM images, but this relationship is difficult to observe from simple biomass versus elevation graphs. However, representing values from each plot as a different dimension in $\mathbb{R}^n$ space, we have found small angles ($[30^\circ - 32^\circ]$ in our dataset) between the normalized elevation vector and the biomass vector. This suggests a nonlinear relationship between these two quantities and we take this as motivation for further computational analysis.

Now, to apply any data-hungry models like deep learning to estimate biomass from these images, one of the main obstacles is the extremely low number of available samples (~ 100) for training and testing. One of the obvious ways to overcome this drawback is to figure out a suitable data-augmentation strategy. In this chapter, we have devised a novel, simple and effective randomized data augmentation scheme that can be utilized to generate a sufficiently large number of augmented samples from each image. The idea is based on swapping similar superpixels in the image randomly. We call this approach the *randomized minimal region swapping (RMRS)* algorithm. The steps of the *RMRS* algorithm are as follows:

1. Get the list of K superpixels from RGB to gray-converted image and sort by their mean values.

2. Generate a randomized list of length N of the number of random swaps needed to generate the pool of N augmented samples from a single image. The random integer values are in the range $[low, \lfloor K/2 \rfloor]$, where low is the predefined threshold for the minimum number of swaps needed to create an augmented sample.

3. For each number r in the list generated in step 2, generate a randomized list of length r of either even or odd superpixel indices in the range $[1, \lfloor K/2 \rfloor]$ and swap minimal rectangular regions between those even(odd) superpixels and their consecutive odd(even) counterparts in the sorted list. Even-odd consideration is necessary to avoid unaugmentation by repeated swaps.

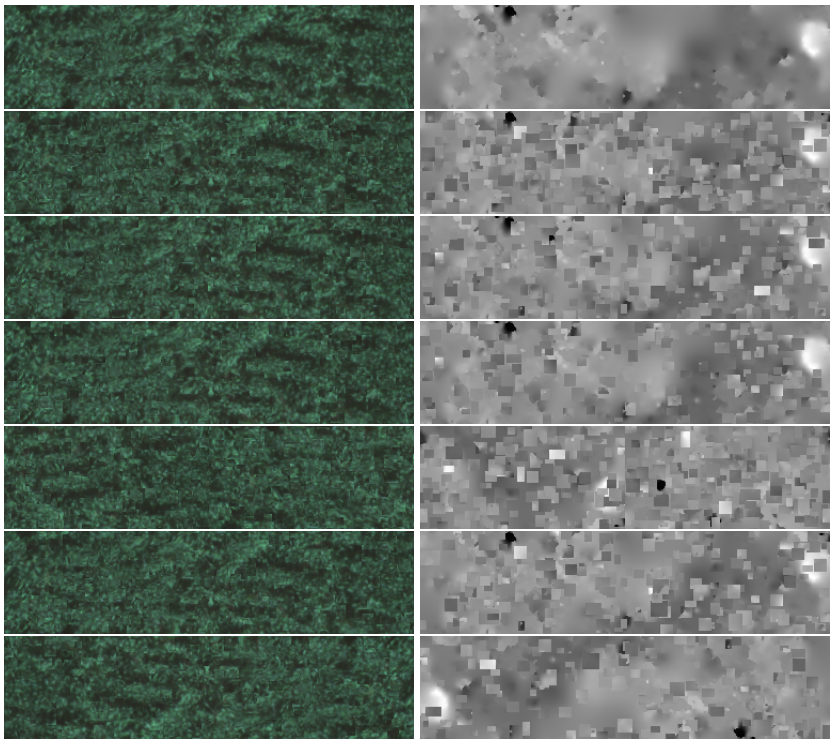

Figure 3.6: Sample RGB plot images (left) with corresponding DEMs (right) showing the original image (top row) and images generated by our *RMRS* data augmentation procedure (other rows). DEM values (height) converted to grayscale for visualization.

In our implementation, we use SLIC [6] as the superpixel algorithm. Figure 3.6 shows sample augmentation results for a single image along with the original one. As can be seen, it is impossible to identify the augmented samples as the artificial ones by looking only at RGB images, even though the corresponding DEMs appear to be highly discretized. Hence, as part of a further exploratory analysis, we plot the normalized summation of all pixel values or elevations of each DEM file for all the augmented samples along with the original one. Figure 3.7 shows this normalized elevation plot for a single image and its augmented samples. As you can see, the normalized elevation varies in the range $[\sim 0.99, 1.0]$, which means that although the augmented DEM files look different and discretized, the contents of the DEM pixels remain nearly constant after being augmented by the *RMRS* algorithm.

In addition to increasing the number of training samples, augmenting data this way has another advantage as a byproduct. We hypothesize that the spatial relationships among the pixels in DEM images have little to do with the prediction of biomass since plants can be found in almost any region in the plot images. Therefore, the counting model should learn to map the pixel values from DEM images into the real-valued space of biomass in an almost spatially invariant manner. For data augmentation by *RMRS* algorithm, new samples are just different permutations of the original one. From the practical standpoint, the interpretation might be that to generate an augmented sample, we swap the plants with similar color information within the plot. Thus, by learning to predict from this augmented dataset, the model may intrinsically learn a spatially invariant mapping from color and elevation to biomass.

Finally, we use a similar network architecture for biomass estimation (Figure 3.4). The only difference between this

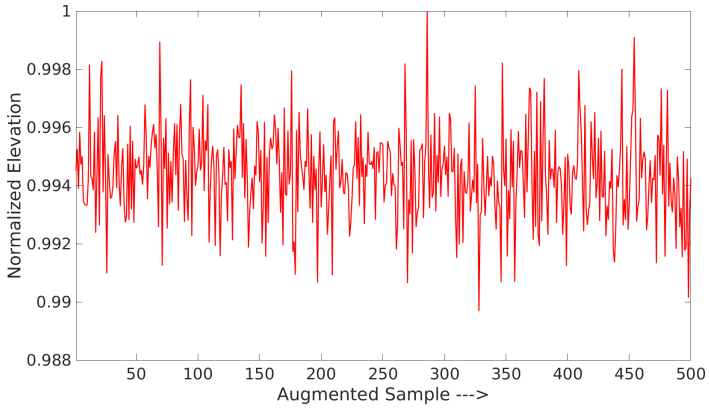

Figure 3.7: Normalized summation of the elevation for the samples augmented from a single image. The first point represents the elevation of the original sample and the rest (499) are the augmented ones. The range of normalized elevation is in the range $[\sim 0.99, 1.0]$ indicating that the total elevation for all the samples are similar to the original.

model and the emergence count one is that the parameters and the placement of the computational blocks or layers are slightly modified to fit the model into this problem.

3.4 Experiments

This section contains the experimental details of our work. First, we describe the datasets used for both tasks. Next, training procedure and implementational details of the networks are provided. Finally, the evaluation metrics are described and the evaluation results are reported in comparison to previous work along with the qualitative visualization of the salient regions.

3.4.1 Datasets

The dataset used for emergence count consists of 274 wheat *(Triticum durum)* plots of $1.5m \times 3.7m$ area. High-resolution aerial images ($\sim 2500 \times 7500$ pixels per plot) were captured for each plot by walking through the field with a GoPro Hero 5 camera [3] mounted on a monopod with a gimbal for stabilization. Covering plots with this device has the advantage of getting very high-resolution images appropriate for detailed computational analysis compared to other remote sensing technologies.

For biomass estimation, we used aerial drone images for 48 wheat (Triticum aestivum) plots for two dates: June 27 and July 20, 2016. The UAV images have been captured using a MicaSense RedEdge camera [4] on a DraganFly Commander drone [2]. The RedEdge camera includes five different sensors, one for each band: $Blue(\sim 465 - 485nm)$, $Green(\sim 550 - 570nm)$, $Red(\sim 658 - 678nm)$, $NIR(\sim 820 - 860nm)$, and $RedEdge(\sim 707 - 727nm)$. The output from these sensors was post-processed using the Agisoft Photoscan [1] to generate an orthomosaic image and digital elevation map. For each of these dates, manual ground truth measurement of biomass have also been conducted. For manual counting, plants were cut randomly from the plots at ground-level using sickles, dried, and then weights of

those plants were noted. The dataset is randomly split into two equal subsets for training and testing.

3.4.2 Training and Implementation

We used Torch [27] as the deep learning framework. To train the segmentation network, we generated $0.25M$ sub-samples of size 224×224 from 10 high-resolution plot images. The network was trained for 30 epochs over this augmented dataset. SGD-momentum was used as the optimizer with a fixed learning rate, momentum, and weight decay of $0.01, 0.9$, and 0.0001 respectively, over the training period.

Both the emergence count and biomass estimation networks were trained with similar parameter settings. Adam optimizer [51] was used with learning rate and weight decay both set to 0.0001. Absolute value and Smooth L1 measures [36] are used as the error criteria (loss functions) for training emergence and biomass models, respectively. For emergence network training, we slowed down the training rate later based on our observation of the training statistics. Training for the emergence network was conducted for 100 epochs, whereas the biomass estimation network was trained with different combinations of input channels for 50 epochs with the same initial parameter settings.

Note that the emergence count network was trained on 7855 patches extracted from 37 images and their slightly augmented versions. On the other hand, the biomass network was trained with about $0.15M$ augmented training samples generated by the *RMRS* algorithm from 48 plot samples. Codes, pre-trained models, and datasets are publicly available here. [1]

3.4.3 Evaluation

Here, we provide three evaluations of our approach. First, we assess the performance of our segmentation network for generating relaxed binary segmentations. Next, both emergence count and biomass estimation networks are evaluated based on the metrics listed in Equation 3.1 below. Among these metrics, we take *MAD* and *SDAD* from the leaf counting benchmark [9]. The other is simply a variant of these measures. In addition, we provide CAM visualization for the emergence counting model.

Table 3.1: Binary segmentation results

$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$	85.59
$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$	83.76
$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{All}$	93.76

Emergence evaluation: Precision, recall, and accuracy are measured to evaluate the segmentation network (Table 3.1). Results for precision ($\sim 86\%$) and recall ($\sim 84\%$) are a somewhat low because the ground truth segmentations are

[1] https://github.com/p2irc/deepwheat_WACV-2018

not precise, but loosely defined contours covering all the plant regions in the images. To justify our outputs, we have visually checked almost all the test segmentation results and find almost no plant regions undetected by the network.

$$\begin{cases} a_i, t_i = \text{actual and target counts for } i^{th} sample \\ N = \text{Number of samples} \\ \textit{\%Difference(\%D)} = \frac{\sum_i |a_i - t_i| I_{[a_i - t_i \neq 0]}}{\sum_i t_i} \\ \textit{Mean Absolute Difference (MAD)} = \frac{\sum_i |a_i - t_i|}{N} \\ \textit{Std Absolute Difference (SDAD)} = \sqrt{\frac{\sum_i (|a_i - t_i| - MAD)^2}{N-1}} \end{cases} \tag{3.1}$$

Table 3.2: Evaluation metrics for the emergence count model

Problem	MAD	SDAD	%D
Prev. Leaf Counting [9]	1.62	2.30	-
Plain Architecture	1.13	1.42	27.04
Inception Architecture	1.08	1.38	25.78
Our Emergence Counting	**1.05**	**1.40**	**25.08**

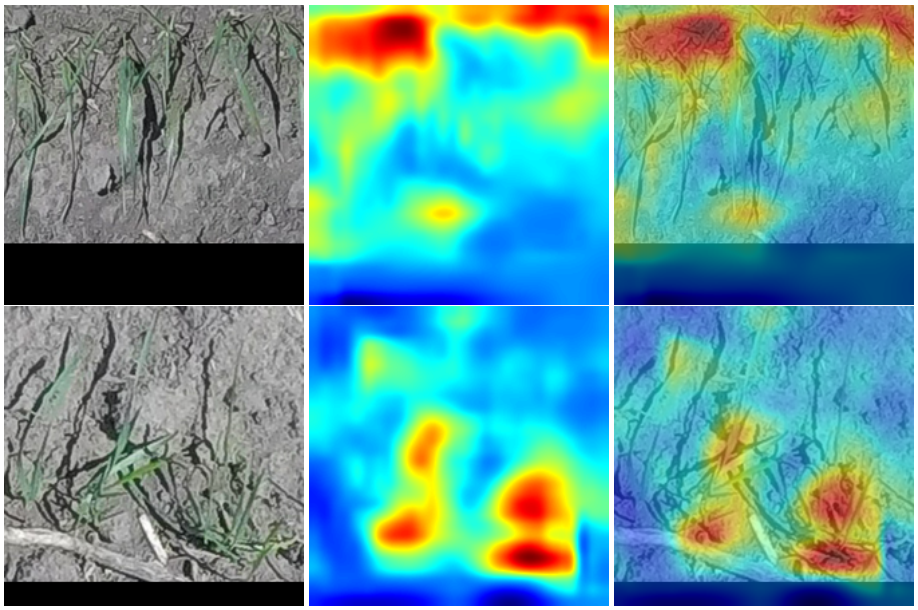

Figure 3.8: Sample RGB images (left), their CAM [122] visualizations (middle), and superimposed images (right). Note that, RGB images are padded by black to maintain a constant size of 224×224. Red and blue indicate the most and the least significant regions responsible for emergence counting. As you can see, the plant bases are detected as the most salient regions (red) which the experts also use for counting followed by the leaves (yellow).

Table 3.2 lists the evaluation metrics for our emergence counting network. As stated in the introduction, we did not find appropriate literature to benchmark our approach. The closest approach is the one used for Arabidopsis and Tobacco leaf counting problem [9]. We achieve *%D* of 25% and *MAD* and *SDAD* of 1.05 and 1.40 which is more accurate than

previously reported results for one of the best leaf counting system currently available. These results are notable because counting wheat plants with thin, overlapping leaves from outdoor images is substantially more difficult than counting leaves from indoor images of rosette plants (as discussed in the Introduction and illustrated in Figure 3.1). We have also included the results for the corresponding plain and Inception-only version to justify the additional complexity of our final model. The plain network was trained for twice the number of epochs than others.

The salient regions detected by our counting model for sample RGB images are shown as heatmaps, generated by CAM [122], in Figure 3.8. Although in the original paper, CAM is used to visualize class-specific mapping of the salient regions, for our counting task, it can also be used for visualizing the regions responsible for making the counts. As already discussed, the bases of leaf-clusters are the most significant parts for successful counts followed by dense regions of overlapping leaves. The sample heatmaps also follow this counting strategy. In the heatmaps, the bases of the plants are marked with red (highest saliency) followed by the leaves with yellow, which clearly indicates that our model is capable of identifying the correct regions in the images responsible for counting. Nonetheless, our percentage deviation is a bit high because of the inherent difficulty of counting the plants due to severe occlusion and large leaf deformations. To enable CAM visualization, we cut out additional fully connected layers, which had provided a slight performance boost, but the resulting visualization provides more valuable insight into the learning process for plant counting.

Table 3.3: Comparison of biomass estimation metrics to other methods and with different input channels ($H \equiv DEM$, Red, Green, Blue, Near-infrared, and red$E$dge)

Method	MAD	SDAD	%D
H_1+MARS [57]	1.66	2.03	29.61
H_2+PLS [56]	3.86	2.72	68.92
H_2+MARS [56, 57]	1.74	2.07	30.96
OH_3+MLR [17]	1.67	1.63	29.67
Ours ($RGBH$)	1.67	2.05	29.75
Ours ($RGBNEH$)	1.53	**1.62**	27.38
Ours (H)	**1.45**	2.05	**25.88**

Biomass evaluation: Table 3.3 contains the same metrics as in Equation 3.1 for biomass models trained with different input channel combinations. Here, H, R, G, B, N, and E stand for DEM, Red, $Green$, $Blue$, NIR, and $RedEdge$ channels, respectively.

As can be seen, the model trained with only $H(DEM)$ as input gives $\%D$ of $\sim 26\%$, which is $\sim 4\%$ and $\sim 2\%$ lower than the model trained with $RGBH$ and all the channels. At this point, it is unclear whether the deep learning model takes care of any of the RGB texture in the biomass image. Intuitively, although color information or greenness of the RGB image might be important, the texture information is not that significant for biomass estimation. However, there is a high variance in the color information under different weather conditions. For instance, if the weather is overcast, crops will appear dark-green, for sunny weather, it will be yellowish-green, and so on. Another critical issue is that after

augmenting data using *RMRS* algorithm, albeit the very local texture property and the total energy of the images are more or less preserved, semi-local texture property is destroyed. We are not sure whether this lack of semi-local texture causes the network trained with RGBE input to perform poorer than the one with only DEM input. This issue can only be explored further if sufficient raw training samples are available in future.

On the other hand, the fact that the model works better when two extra non-visible wavelengths, such as, NIR and $RedEdge$, are provided along with RGB, is consistent with the plant science literature [92] where vegetation indices extracted from hyperspectral and visible wavelength data are used as strong indicators of photosynthetic measurements of plants. However, the utility of hyperspectral data for biomass estimation is still an open question.

In Table 3.3, we provide a comparison against the recent literature. We implemented the methods described in [17, 56, 57] on our data for comparison. These three papers reported the effect of different feature combinations from the set of simple statistical features based on height and different vegetation indices as the predictor variables for their regression models. In this table, we use the combination of features that performed best on our dataset. H_1, H_2, and H_3 indicates slightly different variations statistical height features and OH_3 stands for the combination of H_3 and *Optimized Soil-Adjusted Vegetation Index (OSAVI)*. Also, *MARS (Multivariate Adaptive Regression Splines)*, *PLS (Partial Least Squares)*, and *MLR (Multivariate Linear Regression)* are different linear and nonlinear regression algorithms. As can be seen, even with such tiny amount of original training data, the best performance of our deep model (trained with $DEM(H)$) is $\sim 4\%$ better than the recent nonlinear regression model for biomass.

3.5 Conclusion and Future Work

In this chapter, we have developed three different deep learning models for segmenting plant regions, counting plants, and estimating biomass from aerial field images. Our results show better biomass estimation accuracy than previous methods and better accuracy for outdoor emergence counting as compared to previous studies of indoor leaf counting. Although we have only evaluated our model on particular species of wheat, we expect that our design methodology allows for generalization of these models to other types of crops with minimal changes. As future work, we plan to evaluate our networks with other crops that have different plant morphologies, such as pulses and oilseeds. We also plan to further investigate if estimation accuracy for these phenotypic traits can be improved with larger datasets in subsequent growing seasons, as well as the use of digital elevation maps together with non-visible wavelengths of light as input for biomass estimation.

4. Semantic Binary Segmentation using Convolutional Networks without Decoders

In this chapter, we propose an efficient architecture for semantic image segmentation using the depth-to-space (D2S) operation. Our D2S model is comprised of a standard CNN encoder followed by a depth-to-space reordering of the final convolutional feature maps. Our approach eliminates the decoder portion of traditional encoder-decoder segmentation models and reduces the amount of computation almost by half. As a participant of the *DeepGlobe Road Extraction* competition, we evaluate our models on the corresponding road segmentation dataset. Our highly efficient D2S models exhibit comparable performance to standard segmentation models with much lower computational cost.

4.1 Introduction

Semantic segmentation refers to classifying the pixels of images or videos according to specific categories of objects or background regions known as stuff [21]. Like many other areas of computer vision, research on semantic segmentation has received a tremendous performance boost with the emergence of deep learning in recent years. All recent semantic segmentation models follow a general encoder-decoder type of architecture where the encoder front-end of the network extracts the features necessary for a particular task, and the decoder back-end of the network approximates the segmentation map from these salient features.

FCN [65] is the earliest example of an encoder-decoder style semantic segmentation network. This architecture is built by converting the fully connected (FC) layers at the backend of traditional image classification architectures like AlexNet [55] or VGG [98] into fully convolutional layers with 1×1 convolution followed by upsampled or fractional convolution or deconvolution to generate the pixel-level segmentation map. Skip connections from the higher resolution layers at the convolutional front-end are added for better information gain or performance in both FCN and U-Net [85]. Next comes the SegNet [14] or deconvolutional network [72] architectures, where for upsampling, max-pooling indices are used with the stack of simple convolutional layers. In SegNet, FC layers or equivalent convolution layers are omitted in order to reduce both the memory requirements and computational complexity of the network. Our network design has some similarities to both FCN and SegNet. First, like SegNet, we do not use any FC layers or their equivalent. Moreover, our network uses 1×1 convolution like FCN, but with a much smaller size. Unlike FCN or SegNet, we do not use any deconvolution operation, rather a rearrangement of the feature grid is done by a depth-to-space operation with negligible computational cost.

Recent state-of-the-art segmentation approaches follow the traditional approach of requiring some sort of decoder

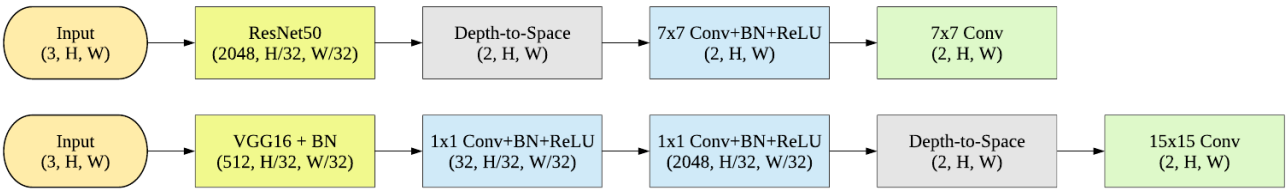

Figure 4.1: D2S models with ResNet50 (top) and VGG16-BN (bottom) backbones. Because of the differences in the shape of the final output layer, the placement of the rearrangement or depth-to-space block is different for these models. The last one or two convolution operations incur a negligible computational cost due to the small number of channels (2).

back-end. The DeepLab models [23] use atrous convolutions [116] in the backend of the CNN models to generate comparatively higher resolution coarse score maps ($1/8^{th}$ of the original image) instead of using max-pooling. They also use spatial pyramid pooling [43] with the atrous convolution of variable rates for better multi-scale prediction as well as fully connected CRF [52] to finetune the bilinearly upsampled score maps. Overall this results in a complex, multi-stage pipeline where CNN and CRF are trained separately, though the latest version of the model [24] omits CRF post-processing. RefineNet [61] uses a multi-path refinement architecture as its decoder. Each refinement block fuses high-level, and low-level feature maps using residual convolution layers and bilinear upsampling for shape adjustment. Also, the authors use the residual sequence of pooling for efficient fusion of multi-scale, pooled prediction. The pyramid scene parsing network (PSPNet) [119] uses pyramid pooling on the feature map of the ResNet equipped with dilated or atrous convolution for global context aggregation. The authors also added a branch in the middle of the ResNet to propagate auxiliary loss for faster convergence. The large kernel paper [75] uses larger convolution kernels to empirically cover larger receptive field [121]. Large symmetric kernels are broken down into a few asymmetric kernels to reduce the computational complexity. Finally, recent literature focused on road mapping from aerial images combine the power of deep learning for pixel-level segmentation with graph-based optimization for the extraction of road topology [29, 71]. All of these related works on semantic segmentation share the common feature of including a decoder sub-network composed of different variations of convolutional and/or upsampling blocks.

In this chapter, we challenge the basic assumption that a decoder sub-network is needed to approximate a segmentation map from encoder-generated feature maps. We hypothesize that, at least for relatively easy segmentation tasks, such as binary segmentation, the computationally-complex decoder procedure can be replaced by a simple depth-to-space rearrangement of the output of the final convolution layer, without loss of segmentation accuracy. We call this type of encoder with depth-to-space (D2S) operation, the D2S network. From an efficiency perspective, our D2S architecture needs only half of the computation to learn the same task.

The idea of depth-to-space reordering that we use in our paper to replace long-range decoders is identical to the sub-pixel convolution for image super-resolution [94]. Depth-to-space operations have also been used before for benchmarking different decoding approaches [110], but in a different way. In that work, multiple instances of the depth-to-space reordering operation are used for 2×2 upsampling in between the convolution layers in the decoder, whereas in our D2S model we use a single depth-to-space block as a replacement for a large stack of convolution layers.

We incorporate our D2S idea as a participant in the *DeepGlobe Road Extraction* challenge. For the competition, our focus is to use a novel and efficient approach instead of an ensemble of sophisticated models. We evaluate our approach on the road extraction dataset. Our D2S model based on the ResNet50 encoder achieves 60.60% mean intersection over union (IoU) whereas the top entry has IoU of 65.60% on the validation set (at the time of writing). This small difference with the best entry validates our hypothesis that for at least easy segmentation problems, encoder-only architectures without any decoder might be a reasonable and efficient model of choice.

4.2 Method

With most current segmentation networks, such as FCN, DeepLab, or RefineNet, the predicted score map has a lower resolution than the input image. For SegNet, the shape of the input image and output segmentation map are the same with the same amount of computation in both encoder and decoder. For that reason, SegNet incurs twice the computational cost of the encoder alone.

In comparison, our architectural design allows pixel-wise prediction naturally. We train the network in such an arrangement where the neighborhood pixel contributions are stored along the depth dimension and then just reordered. One apparent drawback might be that the model will have artifacts in the final prediction map because the contextual mapping task in the neighborhood of the prediction map is interrupted. However, we did not see any such problems in practice, likely because the network learns to overcome the spatial disruption while training end-to-end.

4.2.1 Architecture

We employ Resnet50 [44] and VGG16 [98] with batch normalization [48] as the backbone or encoder of our network with minor differences due to the difference in the dimension of the output of the final convolution layer in these models. The complete architectures for both versions are depicted in Figure 4.1.

We use a similar D2S reordering as proposed previously for the image super-resolution problem [94]. The domain of image super-resolution is a mapping task from the image space to itself with greater detail in the output space. Also, there is no encoder-decoder type of architectures; rather the raw image is taken as a dense feature map and a simple stack of convolution layers are used to produce the corresponding high-resolution version. Thus, the D2S transformation for super-resolution is an arguably more natural operation compared to our case, where we use this block right at the end of the encoder sub-network. In that sense, although we use a similar encoder type of network, it works as a decoder directly from the image space. This decoding task refers to mapping the RGB image pixels into the binary pixel space considering a semi-global context. Theoretically, the amount of context covered depends on the depth of the network. Moreover, we incorporate two-dimensional dropout [105] after each max-pooling for the VGG model and after each block except the last one for the ResNet model for improved performance. For the VGG based model, we use 1×1 convolution to obtain the depth necessary for pixel-level mapping (Figure 4.1, bottom).

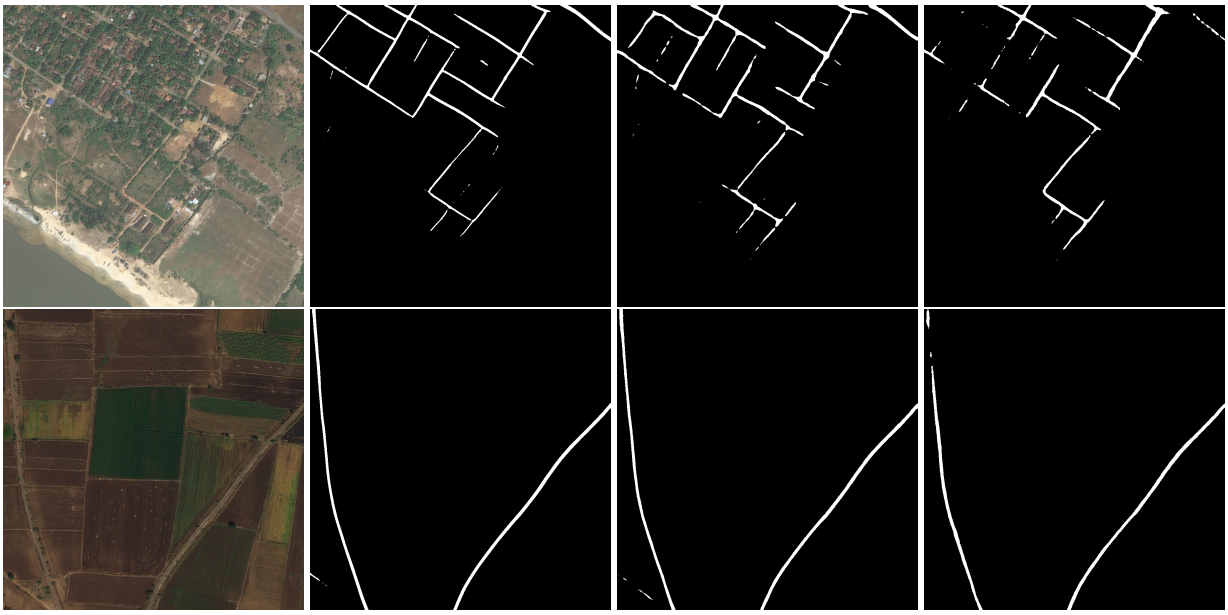

Figure 4.2: (Left to Right) Sample image; Segmentation maps generated by ResNet50-D2S, VGG16-BN-D2S, and Segnet models, respectively.

Table 4.1: Results of our D2S models compared to SegNet as a baseline on the validation set.

Model	**Pixel IoU**
ResNet50-D2S	0.6060
VGG16-BN-D2S	0.5897
SegNet [14]	0.5612

4.3 Experiments

In this section, we provide a brief description of the dataset, the implementation and training details of our models, and results achieved on the validation set in the leaderboard compared to SegNet as a baseline.

4.3.1 Dataset

At the time of writing, the DeepGlobe Road Extraction dataset is only open for the participants of the "DeepGlobe Road Extraction" challenge [31]. The dataset consists of 6226 and 1243 training and validation images, respectively, each of resolution 1024×1024. This dataset is a binary image segmentation problem, where the road pixels are marked as foreground and the rest of the objects and stuff are background. One of the challenges of this dataset is that it is highly imbalanced in terms of the number of pixels per class, i.e. roads are thin lines within the images and therefore occupy few pixels as compared to the background.

4.3.2 Training and Implementation

We train all of our models with ImageNet [86] pretrained encoders. In the beginning, we started training with 224×224 patches extracted from around the true positives in the ground truth due to the scarcity of the foreground (road pixels) in the images. However, empirically we found that having the full context of the image, i.e., training with the whole image at once helped to improve the model's performance.

We use PyTorch [78] as the deep learning framework. All the models are finally trained with full resolution images, and their color jittered versions with the batch size varying in the range of $[3, 8]$. The models are trained on NVIDIA TITAN Xp GPUs and an NVIDIA Quadro P6000 workstation. We use the Adam optimizer [51] with an initial learning rate of 0.0001 which is later reduced based on the training statistics. Codes and pre-trained models are publicly available. [1]

4.3.3 Results

Table 4.1 lists the pixel-level intersection over union (IoU) [31] for three different models on the validation set in the competition leaderboard. We provide the performance metric for a standard SegNet architecture to benchmark our D2S models for a couple of reasons. First, the front-end of our VGG-D2S model is a replica of the SegNet encoder, which is the set of convolution layers of the VGG16 model with batch normalization. Therefore, it is more straightforward to compare the symmetric decoder of SegNet against our spatial rearrangement strategy. Second, SegNet has been reliably employed for binary image segmentation problems with substantial accuracy in recent works [8, 9].

From Table 4.1, we find the D2S models to have comparable performance to the SegNet architecture. Figure 4.2 shows two sample images and their corresponding segmentation maps generated by the three models. From this figure, it is also evident the qualitative performance of the models are quite similar.

Moreover, at the time of writing, the top entry in the leaderboard had IoU of 0.6560 which is $\sim 5\%$ better than our best model. We anticipate that like other featured competitions, the top entries in this competition comprise an ensemble of different approaches, whereas our result is generated using only the D2S models described in this chapter. Therefore, we conclude that for segmentation problems containing only a few classes, heavy-decoder models like SegNet can be reliably replaced by our efficient D2S architecture without significant loss in performance.

4.4 Conclusion

In this chapter, we propose an efficient image segmentation network, called D2S, that uses only a convolutional encoder along with spatial reordering of the final feature maps. Empirically, we show that for relatively easier image segmentation problems, such as binary segmentation, the D2S models give comparable performance to the standard models. Although we only evaluate our model on a simpler problem, this kind of depth-to-space architecture may also be useful in more complex tasks, which we plan to explore in future research.

[1] https://github.com/littleaich/deepglobe2018

5. Preventing False Localization in One-Look Object Counting Models

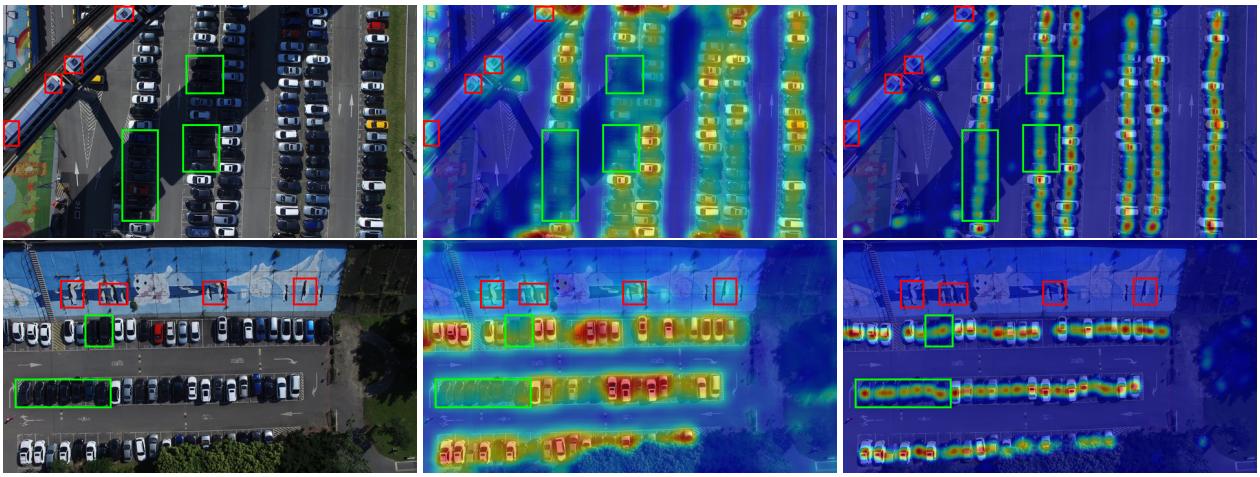

Figure 5.1: (Left) Sample images from the CARPK [46] dataset; Superimposed CAM for the VGG-GAP model trained with only smooth L1 loss (Middle), and joint smooth L1 loss and our proposed heatmap loss (Right). Training with smooth L1 loss exhibits probable false detections for parts of the train and painted wall (red boxes) and probable missed instances for black-colored cars and regions under shadow (green boxes). Without using additional annotations, our heatmap loss fixes both of these problems with more compact activations. Best viewed in digital format.

One-look regression models are widely used object counting models for autonomous vision due to their simplicity. They are trained with weaker dot annotations as count information and are much less computational complex as compared to more sophisticated detection or instance segmentation architectures. Despite reasonable performance, the weak supervision based on a scalar count loss makes these models to miss critical object instances and falsely detect ambiguous background sub-regions; thus resulting in erroneous training of the networks. In this chapter, we propose an efficient loss formulation called "heatmap loss" based on lower resolution CAM and Gaussian activation map (GAM) generated from dot annotations. Training a simple VGG-GAP model with our heatmap loss, in addition to the scalar count loss, improves performance both quantitatively and qualitatively (by suppressing false detections generated by scalar count loss alone) on three heterogeneous datasets – two for car counting (CARPK, PUCPR+), one for crowd counting (WorldExpo), and another for biological cell counting (VGG-Cells). Using the same set of annotations, the addition of our heatmap loss provides $2-3$ lower mean absolute error *(MAE)* count compared to the baseline loss on

these benchmarks and generates state-of-the-art performance on the car park datasets.

5.1 Introduction

Counting object instances from images and videos is a common and practical computer vision task found in a range of applications, such as counting vehicles from aerial images, crowd counting for surveillance, biological cell counting for medical diagnosis, plant counting for image based plant phenotyping, and so on. The problem of object counting can be considered a subproblem of both object detection and instance-level segmentation. However, instance segmentation [41] and object detection [81, 83] pipelines demand ground-truth annotations with much higher specificity as compared to mere object counting frameworks. Obtaining large-scale annotated datasets with fine granularity is a prohibitively time-consuming process. Thus, specialized and computationally efficient counting approaches that exhibit similar performance with weaker image labels (i.e., dot annotations as compared to bounding boxes or pixel-level masks) are worth pursuing for real-time autonomous vision systems where an object count alone is needed.

The predominant deep learning approach for object counting alone is one-look regression, where the model directly predicts a scalar count for an input image. These networks either adopt a classification architecture, where the number of possible output units is a slight overestimate predefined based on the training data [70], or have a single output unit generating a real numbered value as close to the target count as possible [8, 9, 32, 106]. Both variants can be classified under the category of high dimensional nonlinear regression models, where the number of predictors is proportional to the number of input pixels and the number of response variables is either one for the single output case or the highest possible count for the classification case.

Most one-look models are trained on randomly cropped patches, where dot annotations are used to extract the scalar count for the patches on the fly while training. The weaker localization information present in dot annotations is not utilized in the conventional training of one-look models, due to their architectural design and their use of scalar count error based loss formulations, such as L_1, smooth $L1$, and L_2 errors. Even without such supervised localization, one-look models focus, to a certain extent, on the salient regions of the image as indicated by the visualization of class activation maps (CAM) [8, 32, 122]. For counting problems, most of the object instances share similar texture, color, and shape properties in the image space, hence the networks automatically learn to recognize most of them as part of the regression problem.

A fundamental limitation of the lack of localization information in one-look models is that they miss harder-to-detect instances. Consequently, in order to match the count labels in the training samples, these networks compensate for those missed detections by falsely marking a few background sub-regions as object candidates, which have similar spatial properties as most of the true instances. Indeed, we observe this phenomenon when applying a one-look regression model to a recent car counting dataset and visualizing with CAM: dark/shadowed cars are missed and background sub-regions are activated (Figure 5.1). These visualizations demonstrate that posing the counting problem as a mere nonlinear regression problem without further constraints has contextual limitations and can cause an unavoidable generalization error. This is because the only constraint for a weakly-supervised one-look regression network is to map

the input image into a count as close to the target as possible, without explicit information about the spatial position of the target objects in the loss formulation. In other words, it has an abundance of unsupervised spatial context without any prior information about the kind of contextual relationship to be exploited for learning to count. The problem of generating false positives because of missing true positives, in one-look counting models, can only be perceived, and thus avoided, by the network if the loss function is formulated to provide guidance about the probable object locations in the image alongside the scalar target value.

In this chapter, we propose a novel loss formulation based on regulation of the model's CAM heatmap to provide additional contextual information to simple one-look architectures. Our approach does not require additional and/or detailed annotations such as bounding boxes or instance masks and maintains the computational efficiency of one-look models. Alongside the scalar count loss, we backpropagate the difference between the network's CAM and a low-resolution Gaussian activation map (GAM) generated from dot annotations. This weak-error channel can be easily injected into lightweight architectures with a negligible computational expense. To the best of our knowledge, although CAM is widely used to visualize the final saliency map of CNN architectures, this is the first work utilizing CAM directly for any kind of supervision. We evaluate the effect of our heatmap loss on four different object counting datasets — two for cars (CARPK, PUCPR+) [46], one for crowds [117] and the other for biological cells (VGG-Cells) [60]. Training one-look models with the proposed *heatmap loss* alongside the conventional scalar count error loss results in more accurate counts and more compact saliency maps in all cases. Our approach achieves state-of-the-art accuracy on both car datasets and obtains comparable performance on the rest with a simple VGG-GAP baseline model.
Our contributions are summarized as follows:

- We illustrate the unavoidable generalization error that occurs when training one-look counting models using the conventional scalar count loss. This error is caused by the lack of localization information in supervision.

- Without sacrificing the simplicity and efficiency of the one-look models, and using the same ground-truth annotations, we propose a novel loss formulation called *heatmap loss* based on CAM. To our knowledge, we are the first to utilize CAM for loss formulation. We refer to this enhancement for training one-look models *heatmap regulation (HR)*.

- We provide a comprehensive evaluation of our approach on four heterogeneous counting datasets comprising cars, crowd, and biological cells. On all these datasets, additional supervision with the proposed heatmap loss outperforms a conventional one-look model baseline both quantitatively and qualitatively: $2-3$ lower mean absolute error *(MAE)* and more compact final saliency maps. Moreover, the simple VGG-GAP model trained with absolute count loss and heatmap loss performs better or on-par as much more complicated detection or density estimation based approaches on these benchmarks.

- Our approach is more efficient than counting approaches based on saliency or density map estimation with postprocessing [13, 26, 77, 111, 112]. Therefore it is more applicable to real-time and embedded systems where power and computation constraints exist.

5.2 Related Works

Density map estimation: A number of previous works incorporate the concept of density map estimation into the counting pipeline. The most influential early work on counting via density map estimation is done by Lempitsky and Zisserman [60]. They generate pixel-level ground-truth density map from the dot annotations using one Gaussian kernel per object instance followed by density map estimation via linear transformation of the pixel-level feature representations using regularized risk minimization. Their ground-truth density map generation process is similar to our ground-truth activation map generation (Gaussian kernels on dot annotations). However, we used simple per-pixel L_1 metric for the downscaled activation map in our paper which is not suitable for training the models in their framework because of the traditional feature extraction based approach. Extensions to this basic idea are provided in [12, 13, 33, 91, 111]. Fiaschi et al. [33] replaces the linear model of Lempitsky and Zisserman [60] with a regression random forest. The authors in [12] enhance the approach of [60] by providing options to interactively receive and refine dot annotations.

Xie et al. [111] propose fully convolutional regression networks which can be trained on arbitrary input size to predict the final density map. Arteta et al. [13] propose a multi-task deep architecture to predict the density map from noisy crowdsourcing annotations. Segui et al. [91] explore the strength of deep features for one task by training the model on a different task. Encoder-decoder architecture is used in [77] to estimate density map followed by non-maximum suppression to estimate the number of wheat spikes and spikelets.

Alternative spatial map approximation approaches are also proposed in [77, 112]. Xie et al. [112] uses deep learning to predict proximity map instead of density or count map. The Count-ception paper [26] replaces the density map estimation by count map estimation using a fully convolutional network [65] with very small input size(32×32) and then use the idea of redundant spatial coverage to estimate the final count.

Our approach differs from all these density, count or proximity map estimation methods in the sense that our focus is not on estimating any of these maps, instead we emphasize on getting better counting performance by regulating the final activation map or CAM using the Gaussian activation map generated from the simple dot annotations. Moreover, unlike these spatial activation map estimation approaches with long-range and computation-costly decoders, our weak supervision via CAM regulation incurs negligible computational overhead.

Counting networks: Convolutional networks have been tremendously successful in generating region proposals and bounding boxes with associated class probabilities for different categories of objects [81, 83]. The layout proposal network [46] paper enhances the idea of bounding box generation with a domain-specific prior representing the spatial layout of objects. Also, instance-level segmentation is accomplished with substantial accuracy with state-of-the-art convolutional and recurrent network architectures and their hybrids [41, 82, 84]. Although object counts are readily available from these detection and segmentation frameworks, they need more detailed ground-truth annotations which are hard to acquire in a large-scale. Considering this fact, where counting is the only task at hand, convolutional networks are employed as a high-dimensional regression network to generate a real-valued or discrete count from the input image directly [8, 9, 32, 70, 106, 109]. In this chapter, we work on improving the design limitations of these one-look counting models without sacrificing their simplicity and efficiency.

Crowd counting approaches based on traditional computer vision algorithms mostly work on comparatively low-density crowds, whereas recent deep learning approaches exhibit a tremendous performance boost on high-density crowds. Sindagi and Patel [100] provide a comprehensive survey of these approaches. Here, we discuss the ones most relevant to ours. The cross-scene crowd counting paper [117] has a spirit similar to ours in the sense that they also compare their generated density maps with probable Gaussian maps using a deep architecture. The authors use a very small network (only 3 convolution and nonlinear activation layers) and input size (72×72) followed by 3 fully connected (FC) layers. As a result, the network needs to switch between local$-$global$-$local contexts for semi-local density map approximation with the additional risk of overfitting. Furthermore, the process for generating Gaussian density maps for the crowd scenes are dependent on the perspective projection of the camera to take into account the perspective change of the length of the human body in different locations in the image, whereas our Gaussian map generation process is independent of the perspective of the instances. Finally, in the crowd counting paper, additional effort is needed through an extra FC layer to generate the density map and $2-$stage training strategy optimizing the density map estimation loss followed by the counting one. Switching CNN [87] emulates the switching mechanism with a single switching model and multiple regressor models, where the switching network is trained separately to send individual patches into different regressors for optimal performance. Compared to these multi-stage training paradigms, our heatmap loss can be simultaneously optimized with absolute count loss preserving the efficiency of one-look models.

5.3 Our Approach

5.3.1 Conventional Scalar Count Loss

Our motivating hypothesis is that although a simple one-look model naturally learns to localize most target object instances from the image with only scalar count information, the lack of explicit guidance about object properties localization is ultimately detrimental to counting performance. One-look models succeed for most of the instances, except for ambiguous or harder ones. To compensate for the discrepancy in the count compared to the target count caused by these harder instances, the simple one-look model may falsely learn to detect a few background sub-regions as the true candidates which have similar properties as many of the true object instances in the training dataset. Consequently, the one-look regression model suffers from a comparatively higher generalization error. We illustrate this limitation of conventional one-look models by visualizing and regulating CAM as shown in Figure 5.1.

In this figure, we show sample images (left) from the CARPK [46] dataset and its corresponding CAM generated by the simple VGG-GAP model trained with only smooth L_1 scalar count loss and both smooth L_1 loss and our proposed heatmap loss simultaneously. Following the layout proposal network, we remove the last stage of convolutional layers from VGG. For the image in the top row, the model trained with only conventional L_1 loss has a lower emphasis on a few cars under the shadow of the bridge and probably misses some of them. However, it compensates for those false negatives by indicating comparatively stronger activations on the top of the trains, which have similar brightness and texture to white cars in the image. Also for the image in the bottom row, it demonstrates high activations in several spots in the painted side-wall (especially in 4 car-like textured regions). This kind of false detections is hard to avoid using

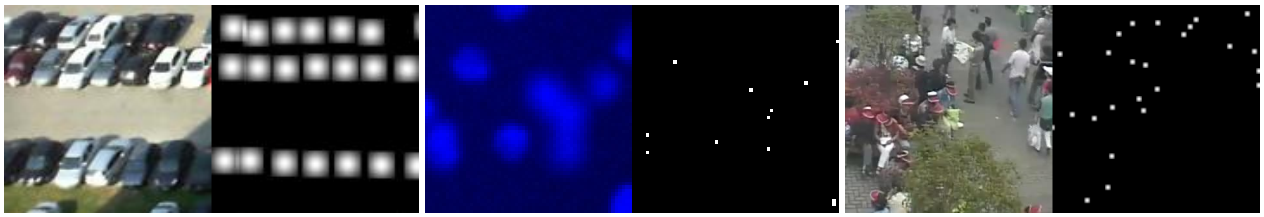

Figure 5.2: Sample cropped images (top) paired with the corresponding ground-truth Gaussian activation maps (GAM) generated from the dot annotations (bottom) for CARPK [46] (left), VGG-Cells [60] (middle), and WorldExpo [117] (right) datasets.

scalar count losses based on L_1 or L_2 errors that only consider the aggregate count information from dot annotations without considering the spatial information while training. This is because the network only knows that it must learn to find a number of similar sub-regions over the whole training dataset, irrespective of the fact that the detected region truly belongs to any of the true objects or not. Consequently, this overly simple learning objective suffers from poor generalization capability. For example, if the trains and the side-walls are inpainted with neighboring road textures in the images in Figure 5.1 and fed to the network as test samples, the network will underestimate the total count even though the new images are similar to their original versions, except for minimal background alteration.

Another drawback of using a scalar count loss is that of the underutilization or suppression of the available ground truth information in training the models. For example, for almost all the counting datasets, the number of object counts are provided in terms of dot annotations. This permits cropping random sized patches and their corresponding counts on the fly while training. The scalar count error function only takes the number of dots inside patches into account and backpropagates absolute differences into the model, completely ignoring weaker localization information provided by the dots. We argue that proper utilization of such additional information would help the model avoiding confusions regarding harder instances and deceptive background sub-regions.

5.3.2 Heatmap Loss

To overcome the drawbacks of scalar count loss, without using any additional or more detailed annotations like bounding boxes or instance masks, we have formulated an approximate and empirical loss based on dot annotations to attenuate such generalization error as much as possible. We extend the idea of CAM [122], which is generated by multiplying the activations of the final convolution layer with the weights of the linear layer following a GAP [62] operation and summing up the weighted responses spatially. During the training phase, we take the network generated CAM as the coarse, lower resolution saliency map of the image and compare it against the probable, downscaled Gaussian activation map (GAM) (Figure 5.2). The GAM is generated as an approximation of the ground-truth activation using a Gaussian kernel with predefined parameters centered on the dots. Our heatmap loss can be written as follows:

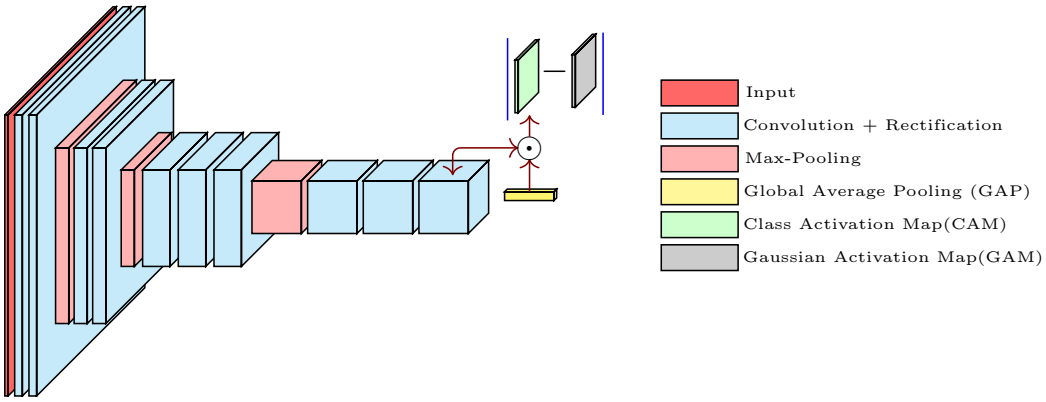

Figure 5.3: We take the first 4 sets of convolution and nonlinear activation layers from VGG16 [98] network, replace ReLU [55] with its parametric version [42], and attach with it a global average pooling (GAP) [62] and a linear layer. Our heatmap loss is the mean absolute difference between CAM low-resolution gaussian activation map (GAM). The double arrow (dark red) indicates that the heatmap loss is backpropagated only through the convolution layers.

$$
\begin{aligned}
\mathcal{G}(x, y; x_c, y_c, \sigma) &= \frac{1}{2\pi\sigma^2} exp\left(-\frac{(x-x_c)^2 + (y-y_c)^2}{2\sigma^2}\right) \\
GAM(x, y) &= \sum_{c:|x-x_c|\leq\omega \,\cap\, |y-y_c|\leq\omega}^{|\mathcal{C}|} \mathcal{G}(x, y; x_c, y_c, \sigma) \\
\mathcal{L}_{heatmap} &= \frac{S^2}{H * W} \sum_{i=1,j=1}^{H/S,\, W/S} |CAM(i, j) - GAM(i, j)|
\end{aligned}
\tag{5.1}
$$

Here, (x_c, y_c) are the coordinates of the dots approximately on the center of the objects, $|C|$ is the number of objects present in the sample, S is the stride of the model, and ω denotes the radius of the *Gaussian* kernel. We empirically set $\sigma = (2 * \omega + 1)/(1.96 \times 1.5)$. We define our heatmap loss $\mathcal{L}_{heatmap}$ simply as the pixelwise L_1 loss between *CAM* and *GAM*. We call this heatmap loss propagation strategy *heatmap regulation (HR)*. Our architecture with *HR* is depicted in Figure 5.3. Note that, following the LPN paper [46], we use the first 4 sets of convolution layers of the pretrained VGG16 [98] network as the convolutional front-end followed by a global average pooling (GAP) and linear layer. Also, we back-propagate the heatmap loss in the convolutional layers only, which is indicated by the double arrow in Figure 5.3.

Intuitively, *HR* is also useful for eliminating the need for nonlinear global information aggregation for counting via multiple fully connected (FC) layers in the front end of the network. For the counting models, where the objects under consideration, like cars, crowd, or cells occupy only a small portion of the whole image, the amalgamation of global information using FC layers should not be necessary. However, fully connected layers might still give better performance for less compact activation maps. Note that, for classification models, typically a gross but somewhat stronger impression about the target object evolved from the vector generated by GAP operation is sufficient to classify the objects in the images with high accuracy. On the other hand, for one-look counting models, comparatively precise activations in the object regions are needed to make the network settle for an accurate count from the vector resulting from the GAP operation at the end of the network. In this case, the learning model might still be able to estimate the

target counts with high accuracy from a less compact activation by means of aggregation of global information via the FC layers as reported in [8]. As already shown in Figure 5.1, the activations generated after conventional training of one-look models are coarse and more distributed over the image than those generated after training with the additional heatmap loss. This dispersed nature of the activation maps might necessitate the inclusion of FC layers for simple one-look models which is not the case for the comparatively compact and concentrated CAM we obtain with heatmap loss. Therefore, we expect that the proposed loss formulation would reduce the need for global decision making via FC layers by enforcing the compactness of the activation maps in the semi-global regions for individual objects. We expect this to improve the overall performance of the models and also prevent overfitting caused by FC layers as a by-product.

Finally, we train our model simultaneously with the heatmap and scalar count losses as follows:

$$\mathcal{L} = \mathcal{L}_{smoothL_1} + \lambda \mathcal{L}_{heatmap} \tag{5.2}$$

Here, λ is the interaction parameter. We set $\lambda = 1$ for car parks and cell datasets and $\lambda = 0.25$ for crowd dataset.

5.4 Experiments

In this section, we provide the experimental results in the form of numerical performance metrics and CAM realizations from both simple and *HR* models. Our models are implemented in PyTorch [78] and will be made publicly available in the final version of the paper. We use the following set of metrics, consistent with previous works:

$$\begin{cases} a_i, t_i = \text{actual and target counts for } i^{th} sample \\ N = \text{number of samples} \\ \textit{Mean Absolute Error (MAE)} = \frac{\sum_i |a_i - t_i|}{N} \\ \textit{Root-Mean-Square Error (RMSE)} = \sqrt{\frac{\sum_i (a_i - t_i)^2}{N}} \\ \textit{\%Underestimate(\%U)} = \frac{\sum_i |a_i - t_i| I_{[a_i - t_i < 0]}}{\sum_i t_i} \times 100 \\ \textit{\%Overestimate(\%O)} = \frac{\sum_i |a_i - t_i| I_{[a_i - t_i > 0]}}{\sum_i t_i} \times 100 \\ \textit{\%Difference(\%D)} = \%U + \%O \end{cases} \tag{5.3}$$

5.4.1 CARPK and PUCPR+ datasets

The CARPK dataset [46] is reportedly the first large-scale aerial dataset for counting cars in parking lots. It contains images with a top-down view covering 4 different parking lots (Figure 5.4, Top). It includes 989 and 459 training and test samples, respectively, each of resolution 720×1280. The total number of car instances in the training dataset is 42274 in the range $[1, 87]$ and in the test dataset is 47500 in the range $[2, 188]$.

The PUCPR+ dataset [46] is published in the same paper as the CARPK dataset. It is a subset of PUCPR dataset [76] which contains images covering a single parking lot with 331 parking spaces of the same resolution (720×1280) as CARPK dataset. The images are captured using a fixed camera from a height of the 10^{th} floor of a building which

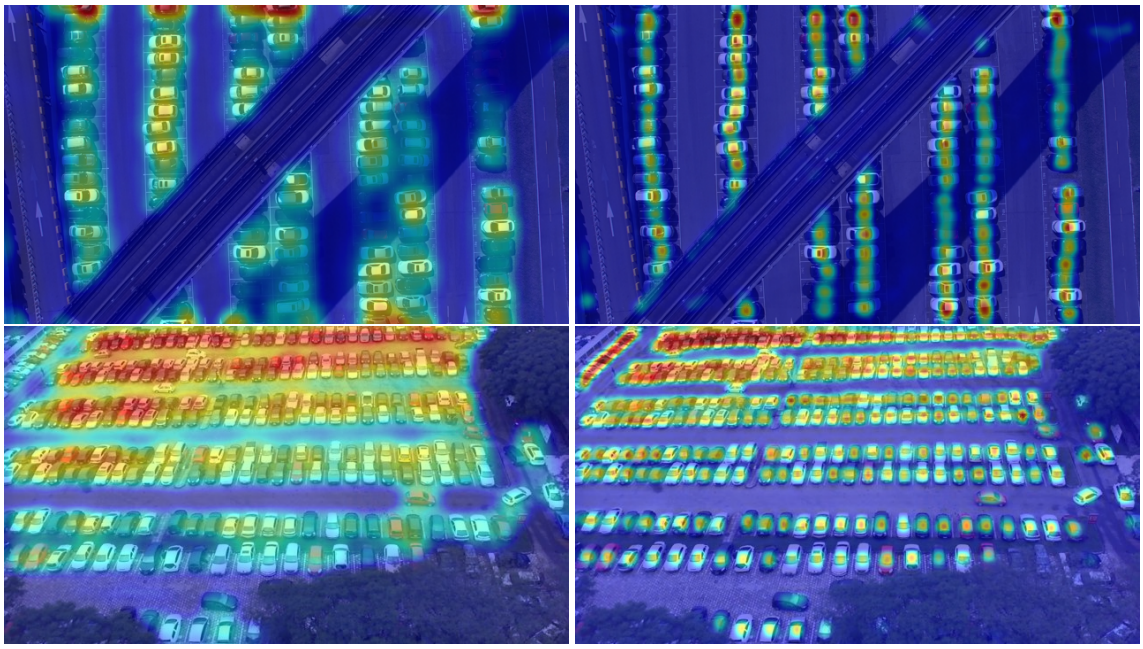

Figure 5.4: Superimposed CAM realizations on the images from CARPK (top) and PUCPR+ (bottom) generated by conventional training (left) and enhanced training with the additional heatmap loss(right). Best viewed in digital format.

provides a slanted view of the parking lot (Figure 5.4, Bottom). This dataset has in total 100 and 25 training and test samples, respectively. The total number of car instances in the training dataset is 12995 in the range $[0, 331]$ and in the test dataset is 3920 in the range $[1, 328]$.

For both car parks datasets, we train the model for $31,300$ steps with batch size of 32 using ADAM [51] with initial learning rate and weight decay both equal to 0.0001. We drop the learning rate to 10% of the initial learning rate after 313 steps and trained both models with that parameter setting for the rest of the steps. Due to similar performance, we train our models on downsampled $(1/2)$ images.

Both models achieve state-of-the-art performance and the *HR* model improves upon the baseline (Table 5.1). Surprisingly, the baseline VGG-GAP model performs better than more sophisticated models like LPN [46] and one-look versions of the large ResNet and Res-ception networks [70]. We expect the success is due to some simplifying aspects of the car counting datasets, including consistent car resolutions, lack of object deformation and lack of intra-object occlusion. However, the datasets do have challenges, such as illumination variance in different parts of the image, occlusion caused by trees and flyovers, and background sub-regions or other vehicles with similar spatial statistics.

Comparing CAM heatmaps for baseline (Figure 5.4, middle row) and HR (Figure 5.4, bottom row), it is evident that the simple VGG-GAP model still places significant emphasis on true object regions. However, the hotspot distribution in the heatmap of CAM generated by the HR model is much more compact than the baseline which is also reflected by its better performance.

Another interesting observation from the car counting results is that the HR strategy helps in suppressing false detections more than in assisting the detection of harder true positives. This can be explained for these particular

Table 5.1: Results on CARPK and PUCPR+ datasets

Method	CARPK					PUCPR+				
	MAE	RMSE	%O	%U	%D	MAE	RMSE	%O	%U	%D
YOLO [46, 81]	48.89	57.55	-	-	-	156.00	200.42	-	-	-
Faster R-CNN [46, 83]	47.45	57.39	-	-	-	39.88	47.67	-	-	-
OLR [46, 70]	59.46	66.84	-	-	-	21.88	36.73	-	-	-
LPN [46]	13.72	21.77	-	-	-	8.04	12.06	-	-	-
VGG-GAP	**10.33**	**12.89**	**1.56**	**8.41**	**9.98**	**8.24**	**11.38**	**0.31**	**4.95**	**5.26**
VGG-GAP-HR	**7.88**	**9.30**	**0.71**	**6.91**	**7.62**	**5.24**	**6.67**	**2.73**	**0.61**	**3.34**

types of datasets. First, most of the false positives are in isolated sub-regions, like train-ends, wall paintings, shadows beneath the trees, etc. Therefore, while training the network with the extra supervision with Gaussian maps, it becomes comparatively easier for the network to suppress these isolated false detections by extracting necessary contextual information. On the other hand, harder true positives are mostly located alongside easier instances. The difficulty comes with low activation in the baseline model due to partial visibility in the border regions, occlusions or different regional statistics, and some of these are confusing even for human vision. Our HR approach succeeds for some of them, such as boosting activations for black cars and cars in shadow, which is evident from the resulting evaluation metrics.

A practical observation we found is that the training error goes down faster for the baseline model than for the HR model, which is consistent with the intuition behind HR. Both the simple and enhanced networks figure out most of the salient regions easily after few passes over the training set, whereas the difference lies in finding the hard cases. For harder instances, the unconstrained loss function based only on the difference in absolute count of the simple model allows it to pick features from a reasonably arbitrary sub-region from the image, be it from the hard, true positives or false background ones. However, HR applies an external force to the model to retrieve information only from the true positive regions, which slows down the convergence speed with the benefit of better generalization performance.

5.4.2 WorldExpo dataset

WorldExpo [117] is reportedly the largest cross-scene crowd-counting dataset, captured using 108 surveillance cameras, each viewing a different scene during the Shanghai 2010 WorldExpo. It has in total 3380 and 599 training and test samples, respectively, each of resolution 576×720. The training and test sets comprise scenes from 103 and 5 different scenes, respectively. In total, there are 182301 person instances in the range $[0, 334]$ in the training set and 42915 instances in the range $[1, 262]$ in the test directories. Both models are trained for $200K$ steps with batch size of 16 using ADAM [51] with initial learning rate and weight decay both equal to 0.0001 with learning rate decayed to 10% after $20K$ steps.

Results demonstrate the superiority ($> 2\%$ improvement) of our HR approach compared to the simple baseline (Table 5.2) with better and more compact activation maps as shown in Figure 5.5. The cross-scene paper [117] achieves better accuracy than our HR approach. However, the overall approach described in that paper suffers from a number of

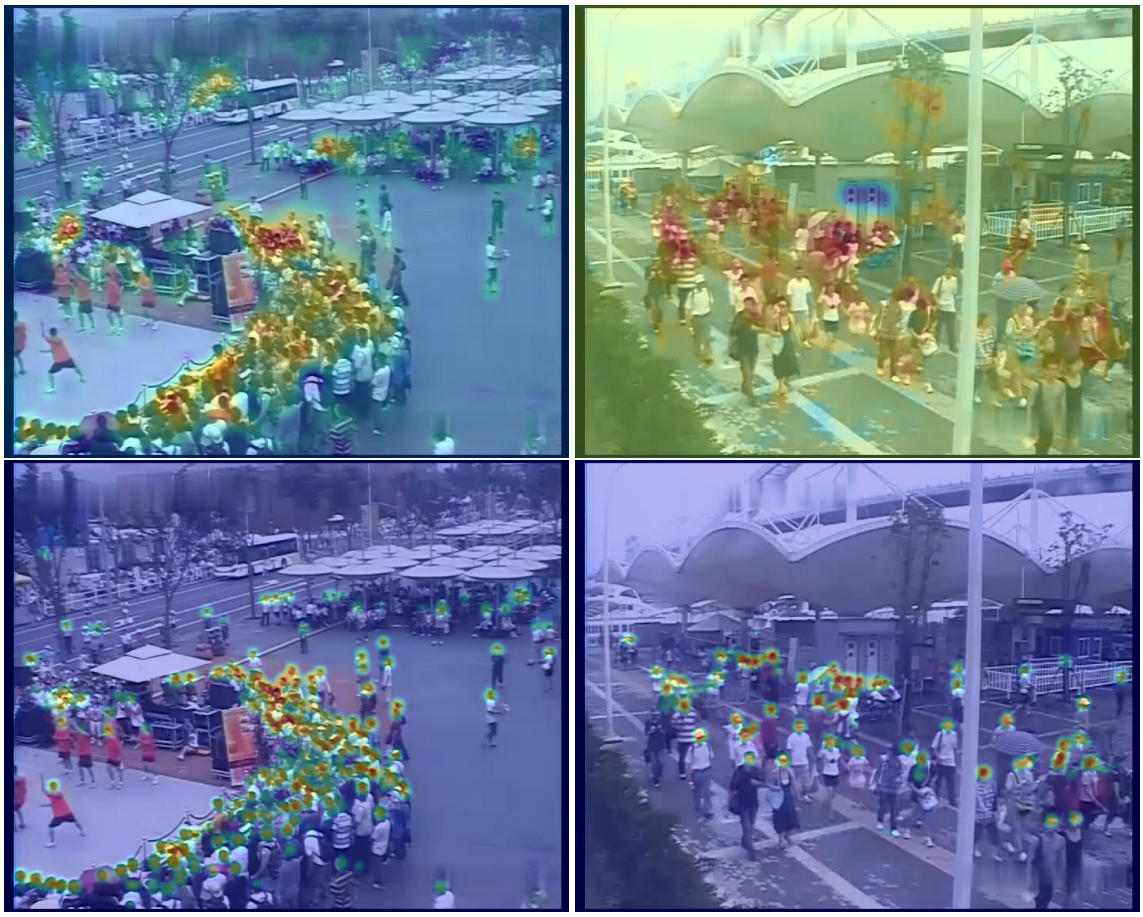

Figure 5.5: Superimposed CAM heatmaps on the sample images for the VGG-GAP baseline (top) and VGG-GAP-HR (bottom) models. Best viewed in digital format.

practical limitations. First, the Crowd-CNN framework is a time-costly and multi-stage process. The CNN model there takes very small patches (72×72) as input and so, the inference on larger images (576×720) incurs time-complexity. In addition, to improve the performance of the model on the test samples extracted from unseen scenes, the authors used fine-tuning or retraining the model already trained on the training scenes using the training patches with similar view angle and scale based on the gradient of the generated perspective maps and crowd density predicted by the pretrained model on the test scenes. In this regard, even though the fine-tuning process is non-parametric, their approach is not readily generalizable to other datasets as a cross-scene crowd counting approach. Finally, the best performance in that paper is achieved in a more cumbersome manner, where the authors first predict the density map over the whole image (576×720) using overlapping 72×72 patches and then use that density map as features for ridge regression (RR) to estimate the crowd density. Also, for switching CNN [87], the authors employed multiple regressor networks with an additional network for switching. All these sub-networks are trained separately in a complex, multi-stage pipeline. Moreover, to retrieve the overall count, smaller patches from each sample need to be inferred with different regressor models based on the decision of the switching network.

Contrary to this heavy multi-stage pipeline, we obtain reasonable performance using a simple network with a

Table 5.2: Results on the WorldExpo test set

Method	Test Directory Name (#Samples, #Count)					Average
	104207 (119, 2168)	200608 (120, 14444)	200702 (120, 9591)	202201 (120, 14050)	500717 (120, 2662)	
LBP+RR [117]	13.6	58.9	37.1	21.8	23.4	31.0
Fiaschi et al. [33, 117]	2.2	87.3	22.2	16.4	5.4	26.7
Ke et al. [22, 117]	2.1	55.9	9.6	11.3	3.4	16.5
Crowd CNN [117]	10.0	15.4	15.3	25.6	4.1	14.1
Fine-tuned Crowd CNN [117]	9.8	14.1	14.3	22.2	3.7	12.9
Crowd CNN+RR [117]	2.0	29.5	9.7	9.3	3.1	10.7
Switching-CNN [87]	4.4	15.7	10.0	11.0	5.9	9.4
VGG-GAP	4.4	26.3	38.9	18.3	7.0	19.0
VGG-GAP-HR	3.3	16.8	23.2	32.0	6.1	16.3

straightforward and fast inference scheme with only one pass over the whole image in a single step without any perspective normalization or fine-tuning for cross-scene adaptation. Also, we generate GAM only around the dot annotations for the heads of people in the image which needs less effort compared to the crowd-counting paper [117], where the activation maps are generated using 2 different Gaussians for both head and body considering their resolution after perspective projection. Nonetheless, it would be straightforward to add heatmap loss into this complicated, multi-stage pipeline and we expect it would further improve the performance.

Table 5.3: Results on the VGG-Cells testset

Method	MAE		%O		%U	
	N=32	N=50	N=32	N=50	N=32	N=50
Learn-to-count [60]	3.5+0.2	-	-	-	-	-
Fiaschi et al. [33]	3.2±0.1	-	-	-	-	-
Arteta et al. [12]	3.5±0.1	-	-	-	-	-
Xie et al. [111]	2.9±0.2	-	-	-	-	-
Count-ception [26]	2.4±0.4	2.3+0.4	-	-	-	-
VGG-GAP	**4.77**	**4.53**	**1.02**	**1.33**	**1.76**	**1.31**
VGG-GAP-HR	**2.95**	**2.67**	**0.76**	**0.93**	**0.96**	**0.63**

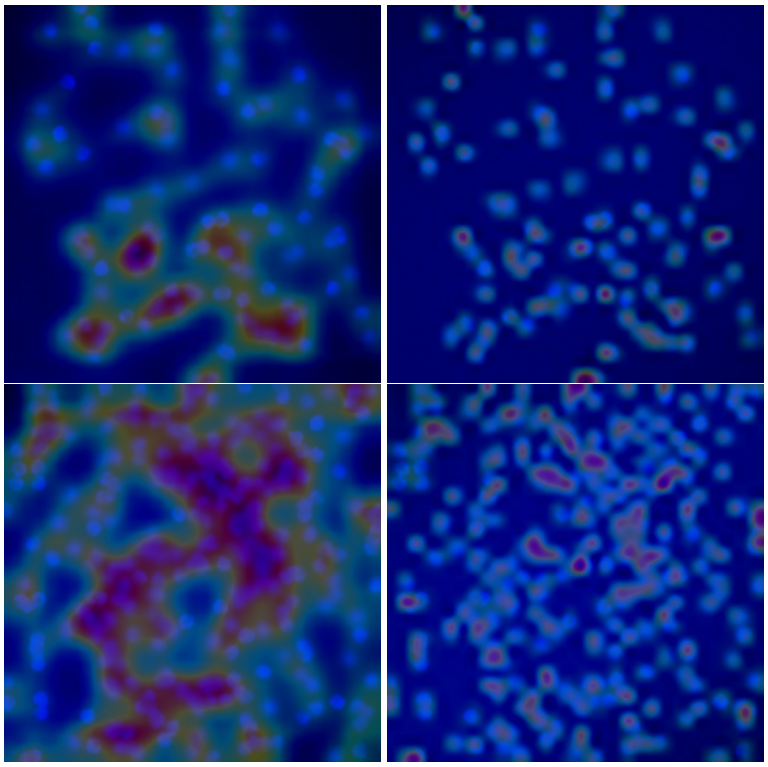

Figure 5.6: Superimposed CAM heatmaps on the sample images from VGG-Cells dataset from the VGG-GAP baseline (left) and VGG-GAP-HR (right) models. Best viewed in digital format.

5.4.3 VGG-Cells dataset

VGG-Cells dataset [60] comprises 200 synthetic images (100 for training and testing each) of resolution 256×256, resulting from the simulation of the colonies of bacterial cells under fluorescence-light microscopy [59]. The total number of cell instances in the training and test sets are 18045 (range [78, 315]) and 17147 (range [74, 317]), respectively. Following the previous benchmarks, we provide two different experimental results: “N=32" for a 32/68 training/validation split and “N=50" for a 50/50 training/validation split. Note that the other methods in Table 5.3 report accuracy for multiple random trials for several random train-val splits, which we think is not appropriate for evaluating deep learning models due to their computational complexity. So, we take the first N samples (32 or 50) from the training sets for training and the rest for validation.

Again we see accuracy gains for HR over baseline (Table 5.3). The improvement in terms of the final activation map over the simple baseline is also evident from Figure 5.6. The Count-ception architecture achieves slightly better accuracy [26] than ours at the expense of more computational cost both at the training and inference stages. They estimate the redundant count maps using a fully convolutional network equipped with more sophisticated Inception-like [104] modules in their architecture. This allows them to obtain multi-scale feature representations with very small input sizes 32×32 to prevent overfitting, followed by redundancy elimination to get the final count. On the other hand, we obtain comparable performance by incorporating our heatmap regulation strategy into a simple VGG-GAP pipeline as before.

5.5 Conclusion

In this chapter, we propose an object-centric loss formulation to improve one-look regression models for object counting without using additional annotations. Our enhancement provides near-to or better than the state-of-the-art accuracy on various counting problems with a simple VGG-GAP architecture containing only ten stacked convolution layers. For the proposed loss formulation, we generate ground-truth activation maps using Gaussian kernels of an approximated average size and predefined standard deviation for each experimental setup, which we find works well in practice. However, estimating these parameters in an unsupervised or a semi-supervised manner from the training dataset might lead to better performance and broader applicability, which we intend to pursue as a future direction of research.

6. Object Counting with Small Datasets of Large Images

In this chapter, we explore the problem of training one-look regression models for counting objects in datasets comprising a small number of high-resolution, variable-shaped images. We illustrate the unreliability of conventional global average pooling (GAP) based models due to the random nullification of true overestimates and underestimates for patchwise inference. To overcome this limitation and reduce overfitting caused by the training on full-resolution images, we propose to employ global sum pooling (GSP) instead of GAP or fully connected (FC) layers at the backend of a convolutional network. Although computationally equivalent to GAP, we show via comprehensive experimentation that GSP allows convolutional networks to learn the counting task as a simple linear mapping problem generalized over the input shape and the number of objects present. This generalization capability allows GSP to avoid both random nullification and overfitting by training on small patches and inference on full-resolution images as a whole. We evaluate our approach on four different aerial image datasets – two car counting datasets (CARPK and COWC), one crowd counting dataset (ShanghaiTech; parts A and B) and one new challenging dataset for wheat spike counting. Our GSP models improve upon the state-of-the-art approaches on all four datasets with a simple architecture. Also, GSP architectures trained with smaller-sized image patches exhibit better localization property due to their focus on learning from smaller regions while training.

6.1 Introduction

Increasingly complex and large deep learning architectures are being devised to tackle challenging computer vision problems, such as object detection and instance segmentation with hundreds of object classes [21, 54, 63]. However, it is becoming common to deploy highly complex state-of-the-art architectures to solve substantially simpler tasks. Object counting is one such task: counting cars on a freeway or in a parking lot, counting people in a crowd, and counting plants or trees from aerial images. While it is possible to apply very powerful instance segmentation [41] or object detection [83] approaches to counting problems, these architectures require detailed (and time-consuming and tedious-to-collect) annotations, such as instance segmentation masks or bounding boxes. However, object counting is amenable to weaker labels, such as dot annotations (one dot per instance) or a scalar count per image. Devising simpler deep learning models for less complex computer vision tasks has the benefit of less costly ground-truth labeling, smaller sized networks, more efficient training, and faster inference.

One-look regression models are a class of deep neural network that are well matched to the comparatively simpler

problem of object counting. These models use a convolutional front-end combined with fully-connected (FC) or global average pooling (GAP) layers that end in a single unit to generate a scalar count of the number of object instances present in the image [8, 9, 32, 106]. Other variants of this counting network use a final classification layer, where the number of the output units are slightly more than the maximum number of possible object instances in the input [70]. This requires that the maximum number of object instances are known *a priori*, which may be difficult when the number of objects varies with the size of the input. Therefore, in this chapter, we focus only on the single output unit models for object counting.

Counting datasets have two common characteristics that complicate the training of one-look models. First, the training set typically consists of a few very high-resolution images. Despite the computational complexity, it might be possible to train on full-sized images as a whole, but there is a high probability of overfitting by blindly memorizing the scalar counts because of the small number of training samples available. Second, images with variable resolution in a single dataset are prevalent because they are often stitched or cropped to a particular region of interest. Many architectures require a pre-defined size for training/test images, and warping aerial images to that pre-defined resolution would make the smaller objects almost disappear or larger objects being unrealistically large. For example, downsampling a crowd counting image makes the instances of smaller resolution even smaller or undetectable and warping to a larger size would result in the crowd closer to the camera being even bigger; thus making the counting problem harder by increasing the ratio of resolution of larger to smaller instances present in the image.

A common solution to overcome the challenge of high-resolution, variable-sized images is to use smaller sized, randomly cropped "patches" from the high-resolution raw training images to train the network. Dot annotations can be counted to create an object count per patch, but because the full extent of the object is not annotated, it is not possible to generate patches without partially cutting the objects at the edge of the patch. This type of label noise may be acceptable during training, but at test time, when a total count is required for the high-resolution image, tiled patches would need to be applied to the network with global average pooling (GAP) layers in the backend and the counts per tile summed. We demonstrate later in the experiments section that using GAP with fixed-resolution, smaller patches incorporates both per-patch underestimates and overestimates. Aggregating the counts of all the patches in a single image randomly nullifies a large number of such overestimates and underestimates; thus giving an apparent impression of a reasonably accurate measure for each image sometimes. However, the extent of this nullification of positive errors by the negatives is random and depends on the pattern of the collocation of the object instances in the images. Also, for rectangular patches, per-patch overestimate and underestimate increase for images with objects not oriented in a spatially vertical or horizontal fashion. We hypothesize that all these shortcomings make the usage of GAP unreliable in the architectures for counting objects with small datasets of high-resolution images.

Previous work has attempted to resolve these patch-wise inference errors empirically, by optimizing the stride of tiled patches based on validation set performance [70]. However, this does not address the fundamental limitation of partial object instances; resulting in unavoidable per-patch counting errors, which are then propagated to the estimate of the full image count.

Considering the complications of datasets with a small number of high-resolution variable-sized images, an ideal

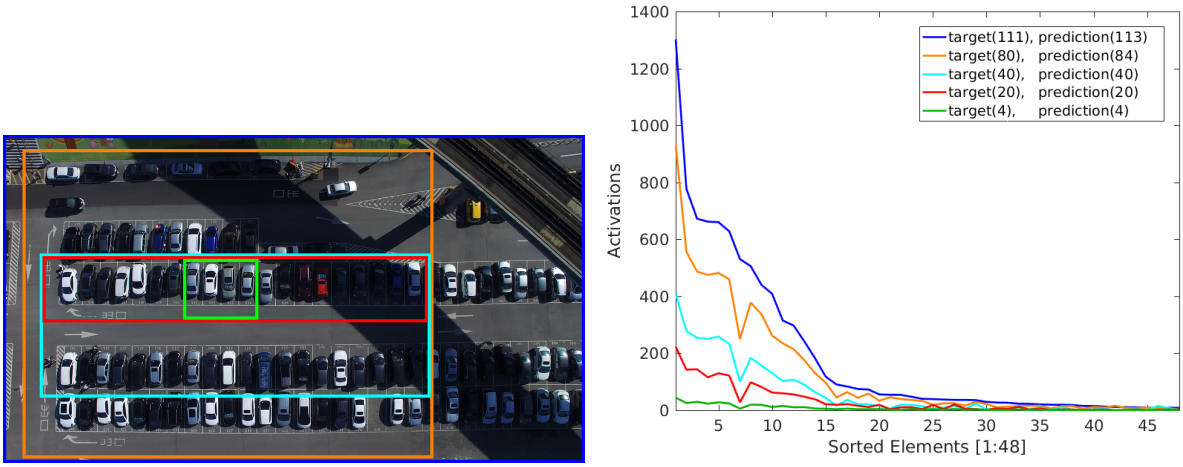

Figure 6.1: (Left) Sample image with multiple cropping shown using bounding boxes with different colors. (Right) Activations of the first 48 elements sorted in descending order incurred by these cropped samples after GSP operation shown using the corresponding colors of the bounding boxes in the left. For consistency, sorting indices of the full-resolution input are used to sort others. The plot of the values demonstrates the fact of learning a linear mapping of the object counts by our GSP-CNN model regardless of input shape.

solution would be a particular kind of model that can be trained with small-sized random patches (to reduce the risk of overfitting or memorization) and then generalize its performance over arbitrarily large resolution test samples. In this chapter, we devise such a model using a set of traditional convolutional and pooling layers in the front-end and replacing the fully connected (FC) layers or global average pooling (GAP) layer with the new global sum pooling (GSP) operation. We show that the use of this GSP layer allows the network to train on image patches and infer accurate object counts on full sized images. Although from a computational perspective, the summation operation in GSP is very similar to the averaging operation in GAP, GSP exhibits the non-trivial property of generalization for counting objects over variable input shapes, which GAP does not. To the best of our knowledge, this is the first work introducing the GSP operation as a replacement of GAP or FC layers. We evaluated GSP models on four different datasets — two for counting cars, one for crowd counting, and one for counting wheat spikes. Our experimental results demonstrate that GSP helps to generate more localized activations on object regions (Figure 6.2) and achieve better generalization performance which is consistent with our hypothesis.

To summarize, our contributions are as follows:

- We describe the limitations of existing architectural designs for object counting on datasets with fewer, high-resolution training samples. With extensive experimentation, we demonstrate the random nullification effect on per-patch overestimates and underestimates of using global average pooling (GAP) with tiled patches. We argue that such randomization makes GAP a fragile candidate for counting architectures.

- We propose global sum pooling (GSP) as an alternative of GAP or FC layers as a remedy to the problems regarding variable resolution images, and overfitting on small datasets of large images. We demonstrate that GSP models can

be trained on smaller random patches and used for inference on full resolution images. This is because the GSP models learn a mapping linear to the number of objects regardless of the input resolution, which is impossible with GAP or FC layer based models.

- We benchmark GSP on four heterogeneous (two car parks, one crowd counting, and one wheat spike counting) datasets. Our GSP model beats state-of-the-art approaches with much better saliency mapping on all these datasets. Results were obtained with a simple convolutional front-end, which demonstrates the simplicity and efficacy of the GSP layer for object counting.

6.2 Related Work

Much of the recent literature on object counting is based on estimating different kinds of activation maps because these approaches are applicable to datasets with high-resolution images. Lempitsky and Zisserman [60] incorporate the idea of per-pixel density map estimation followed by regression for object counting. This regression approach is further enhanced by [12] by adding an interactive user interface. Fiaschi et al. [33] employ random forest to regress the density map and object count. Fully convolutional network [111] is also used for contextual density map estimation irrespective of the input shape. Proximity map, which is the proximity to the nearest cell center, is also estimated in [112] as an alternative to traditional density map approximation. Another variant of density map is proposed in the Count-ception paper [26], where the authors use fully convolutional network [65] to regress the count map followed by scalar count retrieval adjusting the redundant coverage proportional to the kernel size. Wheat spike images have been previously investigated for controlled imaging environments using density maps [77].

There also exists an extensive body of work on crowd counting [100]. Here, we review some of the recent CNN based approaches. Wang et al. [109] employed a one-look CNN model first on dense crowd counting. Zhang et al. [117] developed the *cross-scene* crowd counting approach. They use alternative optimization criteria for counting and density map estimation. Also, instead of single Gaussian kernels to generate a ground truth density map, they use multiple kernels along with the idea of perspective normalization. Cross-scene adaptation is done by finetuning the network with training samples similar to test scenes. Similar to the gradient boosting machines [34], Walach and Wolf [108] iteratively add additional computational blocks in their architecture to train on the residual error of the previous block, which they call layered boosting. Shang et al. [93] use an LSTM [45] decoder on GoogLeNet [103] features to extract a patchwise local count and generate a global count from them using FC layers. CrowdNet [19] uses the combination of shallow and deep networks to acquire multi-scale information in density map approximation for crowd counting. Another approach [118] for multi-scale context aggregation for density map estimation use multi-column networks with different kernel sizes. Hydra CNN [73] employ three convolutional heads to process image pyramids and combine their outputs with additional FC layers to approximate the density map at a lower resolution. Switching CNN architecture [87] proposes a switching module to decide among different sub-networks to process images with different properties.

Most of the approaches described above attempt to approximate a final activation map under different names, i.e. density map, count map, and proximity map, which is then post-processed to obtain the count information; thus resulting

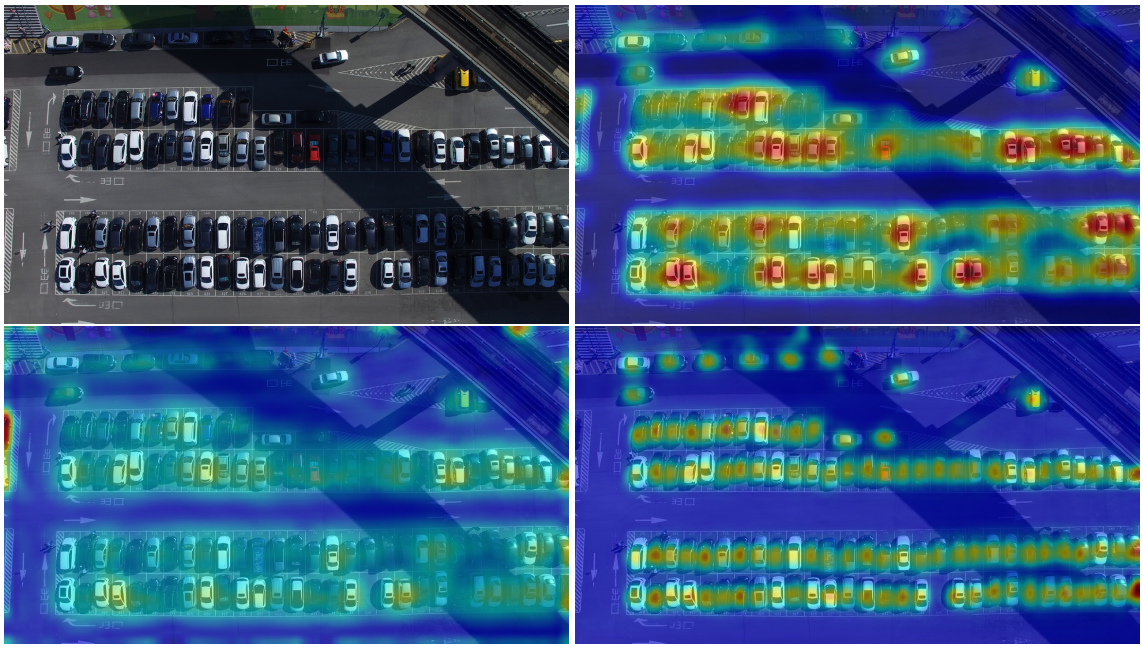

Figure 6.2: Sample image for car counting [46] along with superimposed activation heatmaps for different one-look regression models: (top-left) original image, (top-right) the baseline GAP model, (bottom-left) our GSP model trained with full-resolution images, and (bottom-right) GSP trained with 224 $\times$ 224 randomly cropped patches.

in a multi-stage pipeline. In this regard, one-look models are simpler and faster than these map estimation approaches. The main idea behind using one-look regression models [8, 9, 32, 70, 106, 109] for object counting is to utilize weaker ground truth information like dot annotations, in contrast to more sophisticated models for object detection [81, 83] or instance-level segmentation [41, 82, 84] that require stronger and more tedious to collect ground truth labels. The domain knowledge of spatial collocation of cars in the car parks is exploited in the layout proposal network [46] to detect and count cars. The COWC dataset paper [70] uses multiple variants of the hybrid of residual [44] and Inception [104] architectures, called ResCeption, as the one-look model for counting cars patchwise. During inference, the authors determine the stride based on the validation set. This kind of hybrid models are also used in previous work [8] to estimate plant characteristics from images. However, recent work on heatmap regulation (HR) [10] describes the philosophical limitation of using one-look models and tries to improve its performance by regulating the final activation map with a Gaussian approximation of the ground-truth activation map. In this chapter, using GSP and training with smaller samples, we obtain similar final activation maps to HR without using any extra supervision channel in our model.

6.3 Our Approach

To overcome the generalization challenges for object counting from a small number of high-resolution, variable sized images, and to avoid the problem of partial object counting while cropping random patches, we propose an architecture that can learn to count from images regardless of their shape. Architectures with final FC layers pose strict requirements about the input images' shape, whereas architectures that combine CNN with additional nonlinear and normalization

layers are more flexible. We take inspiration from recent image classification architectures [44, 47, 62, 104] that replace FC layers with a simple GAP layer. Using GAP greatly reduces the number of parameters (to help reduce overfitting), emphasizes the convolutional front-end of the models, permits training and testing on variable size input images, and provides intuitive visualizations of the activation maps [122]. For an object counting task, however, the averaging operation of GAP lacks the ability to generalize over variable resolution input images.

The difference between GAP and GSP for object counting can be illustrated by a hypothetical example. For simplicity of illustration, here we consider an ideal environment where the resolution of object instances falls within a fixed range over a dataset, but this is not a requirement for the GSP approach. Also, without loss of generality, we assume that objects are uniformly distributed, which means that the number of objects within an image is expected to scale with the image resolution. For example, if a $W \times W$ region contains $\mathcal{C}$ objects, then a $mW \times mW$ region would be expected to contain m^2W objects (m is a scalar). If we train a network containing a stack of convolution layers followed by a GAP layer on $W \times W$ samples, our models will learn to generate the expected count of $\mathcal{C}$ with an equivalent vector representation as the output of GAP. During inference, with a $mW \times mW$ image, the last convolution layer will generate m^2 adjacent, spatial feature responses, each representing the expected count of $\mathcal{C}$. This convolutional representation is appropriate to predict an expected count of $m^2\mathcal{C}$. However, the GAP layer will average over all the m^2 spatial sub-regions and obtain an equivalent representation of $\mathcal{C}$. Hence, the averaging operation is not suitable for modeling the proportional scaling of the number of objects with the size of input.

Another option might be to divide the variable resolution input images into fixed size, adjacent, and non-overlapping patches during inference and then sum up the count over all of the filled patches to retrieve the final count. Computationally, such tiling preserves the efficiency of inference in a single pass. Inference on overlapping patches with density estimation is also possible [26], but quadratically increases the computational complexity. Any patch-wise inference scheme, however, has a significant limitation that is unavoidable from the modeling perspective. During inference, the network produces both overestimates and underestimates over all the patches extracted from a single image. For example, for a single image, the amount of overestimates and underestimates are $\mathcal{E}_\mathcal{O}$ and $\mathcal{E}_\mathcal{U}$, respectively. Although the actual difference between ground truth count and prediction is $\mathcal{E}_\mathcal{O} + \mathcal{E}_\mathcal{U}$, by summing up the patch counts, we get an apparent error of $|\mathcal{E}_\mathcal{O} - \mathcal{E}_\mathcal{U}|$. Thus, the measured difference on the whole image, in this case, depends on the difference between overestimate $\mathcal{E}_\mathcal{O}$ and underestimate $\mathcal{E}_\mathcal{U}$, not on their absolute value. For example, we will get a very low error even if $\mathcal{E}_\mathcal{O}$ and $\mathcal{E}_\mathcal{U}$ are quite high but are almost equal. Thus, when aggregating the patch count, overestimate and underestimate get nullified by each other randomly on which the model has no control. We call this effect “random nullification". The amount of such random nullification depends on various properties of the dataset, such as the density and types of the objects, their collocation patterns, their comparative resolution in the image, and so on. In the experiments section, we show that although for GAP models used on adjacent patches for inference, the difference between ground truth and summed up prediction is sometimes reasonably small, the actual overestimate and underestimate are pretty high, validating our explanation of random nullification. Therefore, GAP with adjacent patches is not a reliable solution for inference on variable-sized, high-resolution images.

Instead of average-pooling the final feature maps, we propose a summation or mere aggregation of the input over

Table 6.1: Statistics of the datasets used for evaluation

Dataset	#Images (Train, Test)	Resolution	Total Count (Train, Test)	Range of Count (Train, Test)
CARPK	(989, 459)	(720 × 1280)	(42274, 47500)	([1, 87], [2, 188])
ShanghaiTech-A	(300, 182)	(200×300) – (1024×992)	(162413, 78862)	([33, 3138], [66, 2256])
ShanghaiTech-B	(400, 316)	(768 × 1024)	(49151, 39121)	([12, 576], [9, 539])
COWC	(32, 20)	∼(18k × 18k) – (2k × 2k)	(37890, 3456)	([45, 13086], [10, 881])
Wheat-Spike	(10, 10)	∼(1000, 3000)	(10112, 9989)	([796, 1287], [749, 1205])

the spatial locations only. From the previous example, this aggregation of m^2 similar sub-regions, each with a count of $\mathcal{C}$, would produce the desired expected value of $m^2\mathcal{C}$. Following the nomenclature of GAP, we call this operation global sum pooling (GSP). Although GAP and GSP are computationally similar operations, conceptually GSP provides the ability to use CNN architectures for generalized training and inference on variable shaped inputs in a simple and elegant way. Moreover, due to the single pass inference regardless of input resolution, GSP does not suffer from random nullification.

Linear mapping: Learning to count regardless of the input image shape necessarily means that the convolutional front-end of the network should learn a linear mapping task, where the output vector of GSP will scale proportionally with the number of objects present in the input image. Figure 6.1 shows a sample 720 × 1280 image from the CARPK [46] aerial car counting dataset. On the right of Figure 6.1, we plot the largest 48 activations of the 512-vector output of the GSP layer of our model described later, for different-sized sub-regions of the same sample image. Here, the elements are sorted in descending order for the full resolution image, and the same ordering is used for the activations of the sub-regions. The model producing these activations was trained on 224 × 224 randomly cropped samples. From this figure, it is evident that our model is able to learn a linear mapping function from the image space to the high-dimensional feature space, where the final count is a simple linear regression or combination of the extracted feature values.

Weak instance detector and region classifier: An advantage of training on small input sizes is that it guides the network to behave like a weak object instance detector even though we only provide weak labels (a scalar count per image region). Training on sub-regions of a large input image helps the network to better disambiguate the true object regions from the object-like background sub-regions, resulting in improved performance. For example, when training the network with full images, all of which have a non-zero object count, the network never faces a complete background sample from which it can extract background information similar to any binary region classification problem. On the other hand, when we train with small randomly-cropped regions of the input image many background-only samples are fed to the network, instantiating a more rigorous learning paradigm even with weak count labels. Class-activation map (CAM) [122] visualizations illustrate that the GSP model trained with small sub-regions better captures localization information (Figure 6.2). Training the GAP or GSP models with full-resolution images results in a less uniform distribution of activation among object regions and less localized activations inside object regions as compared to the GSP model trained with smaller patches.

Architecture: We attach a GSP layer after the convolutional front-end of VGG16 [98] model pretrained on ImageNet [86]. GSP produces a 512-dimensional vector, which is converted to a scalar count by a linear layer. We faced no problems with the potential numerical instability caused by large, unnormalized values after spatial summation, even when training the GSP models with full resolution images.

6.4 Experiments

Datasets: We evaluate object counting with GSP on four datasets: CARPK [46] (overhead view of different car parks), ShanghaiTech [118] (crowd images collected from the web and streets of Shanghai), COWC [70] (overhead view of cars in residential areas and highways), and a wheat spike (WS) dataset [50] (overhead view of mature wheat plants). CARPK and ShanghaiTech-B contain constant resolution images, whereas ShanghaiTech-A, COWC, and WS have large images with variable resolutions. All datasets have comparatively few training and test images. Statistics of these datasets are listed in Table 6.1.

Metrics: We adopt the evaluation metrics from MCNN [118] and COWC [70] papers along with one additional metric: the percentage of MAE over expected ground truth, which we call the relative MAE (%RMAE) (Equation 6.1).

$$\begin{cases} \textit{Mean Absolute Error (MAE)} = \frac{\sum_i |\hat{y}_i - y_i|}{N} \\ \textit{\%MAE} = \frac{\sum_i |\hat{y}_i - y_i|}{N y_i} \times 100 \\ \textit{Relative MAE (\%RMAE)} = \frac{MAE \times N}{\sum_i y_i} \times 100 \\ \textit{Root-Mean-Square Error (RMSE)} = \sqrt{\frac{\sum_i (\hat{y}_i - y_i)^2}{N}} \\ \textit{\%RMSE} = \sqrt{\frac{\sum_i (\hat{y}_i - y_i)^2}{N y_i^2}} \times 100 \end{cases} \tag{6.1}$$

Models & Training: We train both GSP and GAP models on full-resolution images and on randomly cropped patches of resolutions 224, 128, 96, and 64. For GSP models, inference is done on full resolution images regardless of their shape and input size used at training, which is not possible for GAP models. For GAP models, we provide error metrics in two forms. First, we report errors over the cumulative patch counts that we denote by *GAP-C (GAP-Cumulative)*. However, as described before, such error is a misinterpretation of the actual per patch error of the GAP models. Therefore, another error is estimated per tiled patch and all the per patch errors over the single image are summed up under the tag of *GAP-PS (GAP-Patch-Summed)*.

In order to train on image patches, we compute a count per patch based on the number of central object regions within the patch. The CARPK dataset provides bounding boxes, which we shrink down to 25% along each dimension and to define a central region for each car instance. The shrinking prevents object regions from overlapping and makes it so that we only count objects that are mostly inside the cropped patch. The ShanghaiTech, COWC, and Wheat-Spike datasets provide dot annotations appropriate to train our models.

CARPK dataset: For this car park dataset, we found that GSP models trained with 128×128 and 224×224 samples perform much better than the same model trained with smaller patches, like 64×64 and 96×96 (Table

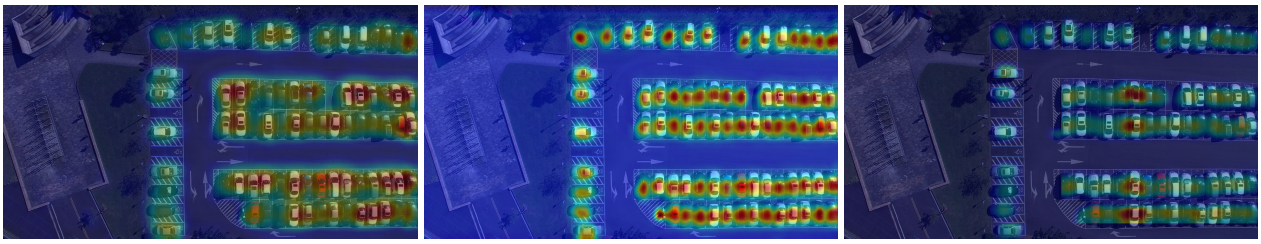

Figure 6.3: Activation maps for CARPK generated by the GAP-Full (left), GSP-224 (middle), and GAP-224(right) models. Activations are more uniformly distributed and more concentrated inside object regions for the GSP-224 model.

Table 6.2: GSP-GAP comparison on CARPK dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
Full	GAP	19.61	21.65	23.78	42.87	18.95
	GSP	32.94	36.23	39.46	70.91	31.83
224	GAP-C	7.65	9.59	9.34	15.65	7.39
	GAP-PS	19.20	21.42	19.01	26.38	16.67
	GSP	**5.46**	**8.09**	**12.21**	**44.36**	**5.28**
128	GAP-C	8.66	11.30	11.90	26.64	8.37
	GAP-PS	22.14	25.94	21.43	34.05	17.88
	GSP	6.70	10.21	8.74	19.15	6.48
96	GAP-C	10.72	13.63	17.23	49.35	10.36
	GAP-PS	44.41	48.49	39.23	54.88	32.89
	GSP	10.63	11.37	22.27	62.87	10.27
64	GAP-C	23.20	27.78	42.16	115.61	22.42
	GAP-PS	52.81	57.24	52.51	110.44	34.98
	GSP	32.09	36.02	31.64	34.39	31.01

6.2). The reason for GAP-C showing apparently lower error is the random nullification of patchwise overestimate and underestimate of GAP models which is evident from the numerics of GAP-PS.

Figure 6.3 compares CAM heatmaps superimposed on original images for the baseline GAP-Full model and our best performing GSP-N model and GAP-N (N=224 for both). The activation maps of the GAP model are variable over the object regions, indicating that some of the objects are being highly emphasized than others, whereas the GSP-224 activations are more uniform, showing that all the instances are getting more or less equal attention from the network. Moreover, the GSP-224 activations better localized within object sub-regions than GAP model, which demonstrates that GSP-N models with small N work as a better object detector or binary region classifier than the baseline models. Moreover, GSP-224 provides state-of-the-art performance (Table 6.3).

We believe that the poor performance of GSP-N models for smaller N (64 and 96) is not a characteristic of the model itself. Instead, the poor performance can be attributed to the training procedure that we followed in this chapter. As already stated, we shrink the bounding boxes for CARPK dataset to disambiguate the overlapping bounding boxes.

Table 6.3: Results on CARPK dataset

Method	MAE	RMSE
YOLO [46, 81]	48.89	57.55
Faster R-CNN [46, 83]	47.45	57.39
One-Look Regression [46, 70]	59.46	66.84
LPN [46]	13.72	21.77
HR [10]	7.88	9.30
Ours (GSP-224)	**5.46**	**8.09**

Table 6.4: GSP-GAP comparison on ShanghaiTech-A dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
Full	GAP	143.13	199.79	43.21	64.84	33.09
	GSP	153.38	259.04	31.98	38.91	35.46
224	GAP-C	83.50	124.39	23.01	34.89	19.31
	GAP-PS	104.13	140.67	28.03	37.37	24.08
	GSP	**70.69**	**103.58**	**19.66**	**29.37**	**16.34**
128	GAP-C	83.75	124.79	23.28	34.27	19.36
	GAP-PS	113.31	148.58	30.63	38.50	26.20
	GSP	71.19	111.86	18.73	29.28	16.46
96	GAP-C	86.81	124.88	23.31	30.71	20.07
	GAP-PS	127.63	162.94	33.35	37.79	29.51
	GSP	78.25	116.69	19.87	27.21	18.09
64	GAP-C	100.69	140.53	26.87	33.80	23.28
	GAP-PS	160.38	197.79	42.16	45.99	37.08
	GSP	107.81	151.72	29.98	37.97	24.93

However, such shrinking poses restrictions on using arbitrarily small sample sizes in training. If the patch size is close to the object resolution (the average resolution of the bounding boxes in the training set of CARPK dataset is about 40 pixels) and the objects are close together (which cars are in a parking lot), a patch is likely to include one complete object with several other instances partially cut at the edge of the patch. Because we disambiguate object counts by shrinking the boxes, depending on the relative orientation between the object and its encompassing box, and its portion inside the cropped patch, it might be taken into account for counting or not. Therefore, this aspect of our training paradigm is a bit randomized. For comparatively larger sample size, such as 128 and 224, we anticipate that this problem of random consideration of the partial objects in the border is less frequent than the smaller sized patches, such as 64 and 96. In this regard, the optimal sample size depends on the average resolution of the object instances and their relative placement in the images of a particular dataset.

ShanghaiTech dataset: This is the largest crowd counting dataset in terms of the number of counts (Table 6.1). It comprises two parts – part A images are randomly collected from the web and part B is acquired from the busy streets

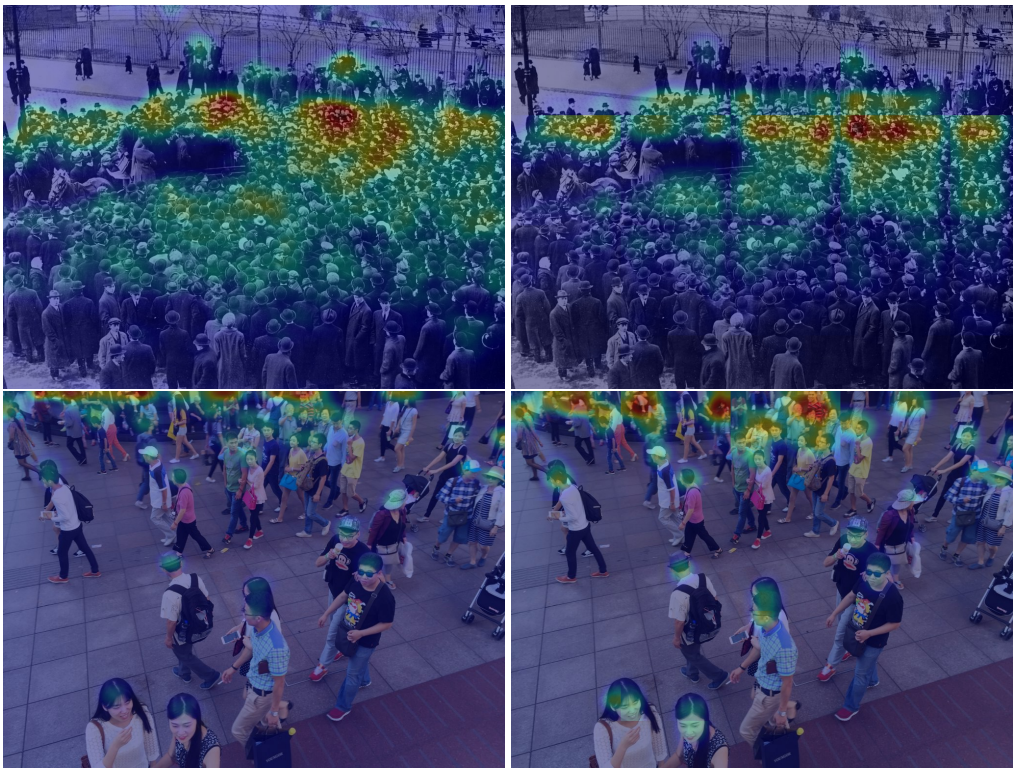

Figure 6.4: (Left) Saliency maps generated by the best GSP models on ShanghaiTech-A (top) and -B (bottom) datasets. (Right) Same for the best GAP models. GSP and GAP models exhibit similar activations, except GSP models are free from *random nullification* effect due to single inference on full image.

of Shanghai. Table 6.4 and 6.5 enlist the comparative performance of GSP and GAP models on these subsets. Table 6.6 reports the comparison with state-of-the-art approaches.

Our GSP-224 (part A) and GSP-128 (part B) models outperform state-of-the-performance approaches. Note that, although GAP models apparently provide good accuracy, their per patch error is pretty high indicating a considerable amount of random nullification. This is also evident from Figure 6.4. In this figure, both the best performing GSP and GAP models show similar saliency maps validating our claim that random nullification is heavily responsible for the comparatively poor performance of GAP models.

COWC dataset: COWC contains very few training images (32) and the image sizes vary substantially (2220×2220 to 18400×18075), therefore it is an ideal test case for the main features of GSP. Unlike the parking lot datasets, the COWC dataset contains images covering highways and residential areas and therefore cars in these images often appear to be entirely isolated objects in the roads or highways or parked in the residential streets. Each pixel covers 15 cm, resulting in the resolution of the cars ranging from 24 to 48 pixels. Because of the sparsity of the objects in the ultra-high-resolution training images, we extract ~ 8000 samples of resolution 288×288 centered on object sub-regions from the images prior to training. We do this to avoid training on a large number of negative samples that would be the case for random cropping.

For COWC, we could not provide per patch error metrics for GAP models in Table 6.7 since the test set contains only scalar counts as the ground truth. The smallest patch size (GSP-64) provides comparable performance to previously

Table 6.5: GSP-GAP comparison on ShanghaiTech-B dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
Full	GAP	12.91	20.19	13.44	21.66	10.45
	GSP	12.26	19.49	12.01	19.56	9.92
224	GAP-C	9.70	16.03	7.69	10.16	7.84
	GAP-PS	18.44	24.04	15.87	17.38	14.91
	GSP	9.96	16.67	8.07	10.71	8.06
128	GAP-C	10.24	16.81	8.43	11.34	8.29
	GAP-PS	24.05	30.31	21.24	22.76	19.45
	GSP	**9.13**	**15.94**	**7.05**	**9.24**	**7.39**
96	GAP-C	11.16	17.60	9.31	11.93	9.03
	GAP-PS	28.73	34.74	25.49	26.76	23.24
	GSP	9.48	15.40	7.70	10.21	7.67
64	GAP-C	15.94	21.57	15.69	18.69	12.90
	GAP-PS	39.06	46.88	35.30	36.75	31.60
	GSP	14.35	22.95	12.44	15.85	11.61

Table 6.6: Results on ShanghaiTech dataset

Method	Part A		Part B	
	MAE	RMSE	MAE	RMSE
CS-CNN [117]	181.8	277.7	32.0	49.8
MCNN [118]	110.2	173.2	26.4	41.3
FCN [66]	126.5	173.5	23.76	33.12
Cascaded MTL [99]	101.3	152.4	20.0	31.1
Switching-CNN [87]	90.4	135.0	21.6	33.4
Ours (GSP-224 and -128)	**70.7**	**103.6**	**9.1**	**15.9**

published results (Table 6.8). We also see that the activations for GSP-64 are more concentrated on the objects in the image compared to that of GSP-224 (Figure 6.5). This observation is consistent with our claim that the GSP models trained with smaller sample size tend to localize objects better, particularly when they are relatively isolated from each other.

Wheat-Spike dataset: This dataset [50] is a comparatively challenging one for object counting because of the irregular placement or collocation of wheat spikes. Out of 10 training samples, we use 8 for training and 2 for validation. Like COWC, the Wheat-Spike dataset is an ideal case study for GSP because of the low number of high-resolution training samples. Since the images are high-resolution and sub-regions inside a single image vary quite a bit in terms of brightness, perspective, and variable object shape resulting from natural morphology and wind motion, there are many features inside a single image that any suitable architecture should exploit without memorization or overfitting.

Table 6.9 reports the comparative performance of GSP and GAP models on this dataset. Although the summed

Figure 6.5: Superimposed activation maps for GSP-64 (left) and GSP-224 (right) on the cropped image of COWC dataset. Activations are better localized the GSP-64 model.

Table 6.7: GSP-GAP comparison on COWC dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
224	GAP-C	17.54	22.98	36.47	50.05	10.15
	GSP	8.85	13.01	10.70	14.99	5.12
128	GAP-C	20.05	43.18	13.89	17.72	11.60
	GSP	8.45	13.09	12.22	17.84	4.89
96	GAP-C	15.45	25.64	10.44	11.99	8.94
	GSP	**8.20**	**12.53**	11.13	16.38	**4.75**
64	GAP-C	24.34	45.16	19.30	24.09	14.09
	GSP	11.15	23.61	**5.72**	**8.43**	6.45

up count for GAP seems to be more accurate than the corresponding GSP models, the surprisingly high aggregate of per-patch error again explains the effect of random nullification on patchwise inference with GAP models.

Also, the error for the GSP model trained with full-resolution images is quite high – 161.63, about 16% *MAE* compared to the average count of 1000. GSP-96 provides the best performance with MAE of 80.00 (8% of average count). Figure 6.6 shows cropped samples, their superimposed activation maps from GSP-Full model (middle), and GSP-96 (right). The GSP-96 model is able to identify salient regions in the image well, but for GSP-Full models, it tries to blindly memorize the count from only eight high-resolution images, which is clearly evident from the very uniform heatmap distribution all over the image regardless of foreground and background.

Table 6.8: Results on COWC dataset

Method	%MAE	%RMSE
ResCeption [70]	5.78	8.09
ResCeption taller 03 [70]	6.14	7.57
Ours (GSP-64)	**5.72**	8.43

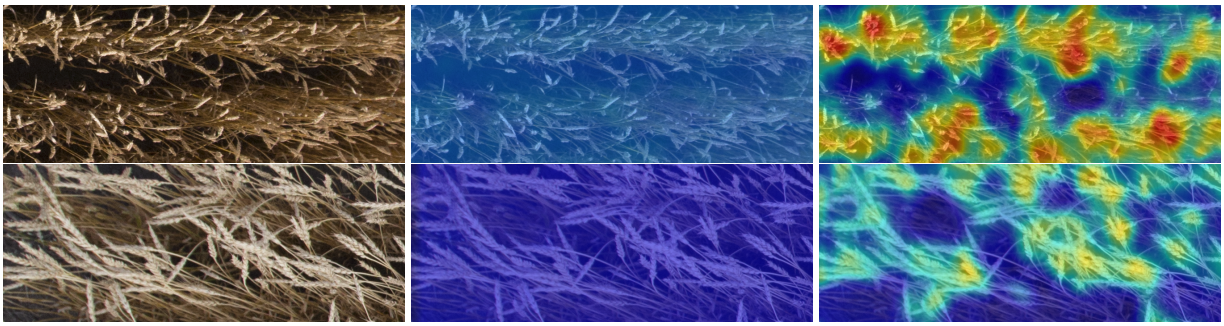

Figure 6.6: Cropped sample images from Wheat-Spike dataset (left) with superimposed CAM generated by GSP-Full (middle) and GSP-96 (right) models.

Table 6.9: Results on Wheat-Spike dataset

Input	Type	MAE	RMSE	%MAE	%RMSE	%RMAE
Full	GAP	132.25	153.77	13.82	16.40	13.24
	GSP	161.63	178.11	16.16	17.81	16.18
224	GAP-C	82.19	92.41	8.43	9.45	8.23
	GAP-PS	189.5	195.06	17.94	18.36	17.85
	GSP	108.19	134.01	10.37	12.38	10.83
128	GAP-C	91.00	106.07	9.45	11.07	9.11
	GAP-PS	279.75	284.67	24.93	25.08	24.92
	GSP	85.00	108.87	8.05	9.95	8.51
96	GAP-C	**75.38**	**88.23**	7.83	9.37	7.55
	GAP-PS	351.50	356.98	30.34	30.52	30.26
	GSP	80.00	100.63	7.94	9.93	8.01
64	GAP-C	87.50	99.02	9.19	10.46	8.76
	GAP-PS	514.00	519.63	41.29	41.49	41.07
	GSP	111.38	130.07	11.23	13.01	11.15

6.5 Conclusions and Future work

In this chapter, we introduce the global sum pooling operation as a way to train one-look counting models without overfitting on datasets containing few high-resolution images. With detailed experimental results on several datasets, we show that our GSP model, trained with small numbers of input samples, provides more accurate counting results than existing approaches. Also, when the GSP model is trained on small patches, it indirectly receives a weak supervision regarding object position and learns to localize objects better. This is also true for a GAP model, but it lacks the capability to infer counts from full-resolution test images and suffers from *random nullification* for patchwise inference. This makes GAP unreliable for object counting on variable-sized, high-resolution images. Although we have only addressed object counting in this study, we believe that GSP could be applied to other computer vision tasks, such as classification or object detection. For these tasks, we expect that the scaling property of GSP may be able to utilize the features of image sub-regions over multiple spatial scales better than models that employ GAP or FC layers. In that case,

the requirement for fixed-resolution images for many object detection or classification models can be eliminated. We plan to investigate these directions as future work.

7. CONCLUSION

In this work, we focused on several aspects of building a general-purpose object counting system based on the popular one-look ConvNet models. We also made strides toward improving the efficiency of semantic segmentation architectures for simpler problems that might appear as a subroutine of the object counting algorithm. Although the thesis is mostly based on deep learning, it is not devoid of conventional computer vision and image processing techniques, which significantly assisted in the crucial data augmentation stage; thus making the training with small datasets feasible. Below we provide a brief summary of this work and cast light on the possible future directions for investigation.

7.1 Summary of Contributions

- **Chapter 2:** We dived into the research on object counting by building a simple ConvNet architecture from scratch on a leaf-counting dataset. The dataset is somewhat easier in the sense that it contains images acquired under a controlled, indoor environment. Unlike recent works on the same dataset deploying specialized models for images of various types (different resolution, background, and species), we use a single, moderately deep model to provide counting performance in a generalized manner.

- **Chapter 3:** Next, we followed the same principle on another dataset for estimating the density of wheat plants in the early-season images captured in an unconstrained, outdoor environment. This dataset contains thin structures (wheat leaves) and severe occlusion, resulting in ambiguous ground truth annotations. We took a divide-and-conquer approach where we crop the small patches from high-resolution images with the idea of connected-component labeling and aggregate the counts afterward. However, we had to incorporate more sophisticated architectural design concepts for better convergence on this harder dataset. In addition, for biomass estimation, we proposed an efficient, randomized data augmentation algorithm based on a conventional computer vision method (superpixel) that made the training of standard ConvNets on a tiny dataset with low-resolution images feasible.

- **Chapter 4:** After that, we investigated an approach for simpler semantic segmentation since this appears to be an essential component of the counting pipeline sometimes depending upon the nature of the problem and the datasets. Emphasizing our motto of simplicity and efficiency, we propose to use depth-to-space reordering of the feature elements in the very deep convolution layers of the network to form the segmentation map. Although depth-to-space was used before for image super-resolution, which can be considered a sophisticated upsampling or homogeneous mapping task, our work is the first to use it for any kind of heterogeneous mapping (RGB pixel space to binary object space).

- **Chapter 5:** Next, we illustrated a subtle problem of training such one-look models with a simple scalar count loss. Despite the outstanding ability of the models to identify most of the object instances from the images with only count information, confusions and mispredictions arise for harder true positives as well as deceptive background sub-regions. To alleviate this problem, we propose a weakly-supervised loss based on the dot annotations already available to the counting datasets and predicted heatmap of the model. Our auxiliary loss was shown to provide better performance both quantitatively and qualitatively across various counting datasets.

- **Chapter 6:** Finally, we elaborated the issue of uncontrolled errors evolved from the tiled, overlapping prediction conventionally used to deal with variable-shaped, high-resolution images. As a remedy, we proposed the trick of using global sum pooling (GSP) that simply bypasses the option of tiling as well as the uncontrolled, random error.

7.2 Possible Future Directions

Here, we restate the future works listed at the end of each of the manuscript-style chapters before.

- **Chapter 2:** We segmented the target region prior to counting with the assumption that such segmentation should assist the model to further emphasize on a reduced search space and provide better performance. However, recent approaches exhibit better performance working directly on the raw images including background. From our perspective, one possible reason for such discrepancy is that the model easily overfits on the low dimensional manifold. Therefore, delving deeper into the need for prior segmentation for different fundamental vision tasks, including counting, is a fascinating empirical direction for future research.

- **Chapter 3:** This chapter focuses on extending the idea of building deep models from scratch on outdoor images with small number of training samples. We obtain reasonable performnace on a single species of crop, i.e. wheat. However, many plants/crops have roughly similar morphological characteristics, which ConvNets are supposed to capture well. Hence, for the species lacking ground truth, being able to train on the available images of a set of similar plant species and use for inference on the new types would remove a huge bottleneck from the applications perspective. Furthermore, any kind of quantification of the transferability of knowledge of the models trained on different species should prove to be a self-rewarding future direction.

- **Chapter 4:** As the title of the chapter says, we experiment on the idea of depth-to-space reordering for efficient semantic segmentation on simpler cases like binary segmentation. Therefore, an obvious future direction is to extend this idea for multi-class segmentation. Moreover, an in-depth qualitative analysis on the possible checkerboard effect due to dimension-shift will help providing an empirical bound on smoothing afterward.

- **Chapter 5:** To generate the approximate density map for heatmap loss in Chapter 5, we estimate the expected resolution of the object from the training set via initial exploratory analysis. In addition, we empirically set another hyperparameter, sigma of Gaussian kernel, in the experiments. We believe that semi-supervised or unsupervised approximation of such hyperparameters would make the overall approach even more robust. Another shortcoming

of our straightforward formulation is the lack of adaptability to highly variable resolution of objects in the images, where again weak-supervision could be useful. Future work should point towards the enhancement of our preliminary setup for such non-trivial cases.

- **Chapter 6:** The GSP operation proposed in Chapter 6 overcomes the limitation of somewhat uncontrollable error produced by the ConvNets in case of tiled approximation. Also, it forms a linear relationship between the object count in the image and the value of the final FC layers in the model regardless of the resolution of input. This kind of generalized linearity induces a future direction for investigating GSP for other kinds of vision tasks as well, such as multi-scale object classification, detection, segmentation etc.

7.3 Final Remarks

As a concluding remark, this work has a limitation of the lack of theoretical proofs as the means of justification of the proposed ideas in its entirety, which might be partially credited to the theoretical incapability of the student himself during writing. However, all the ideas presented here are thoroughly experimented across multiple datasets from the appropriate sub-domains of object counting. That said, we hope this work will provide empirical guidance to both theoretical and practical enthusiasts interested in deep learning based object counting research, in general.

REFERENCES

[1] Agisoft. http://www.agisoft.com/.

[2] DraganFly. http://www.draganfly.com/.

[3] GoPro. https://gopro.com/.

[4] Micasense-Rededge Camera.
https://www.micasense.com/rededge/.

[5] Leaf Counting Challenge.
https://www.plant-phenotyping.org/CVPPP2017-challenge, 2017.

[6] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE TPAMI*, 2012.

[7] R. Adams and L. Bischof. Seeded region growing. *IEEE TPAMI*, 1994.

[8] S. Aich, A. Josuttes, I. Ovsyannikov, K. Strueby, I. Ahmed, H. S. Duddu, C. Pozniak, S. Shirtliffe, and I. Stavness. Deepwheat: Estimating phenotypic traits from crop images with deep learning. In *IEEE WACV*, 2018.

[9] S. Aich and I. Stavness. Leaf counting with deep convolutional and deconvolutional networks. In *IEEE ICCVW*, 2017.

[10] S. Aich and I. Stavness. Improving object counting with heatmap regulation. *CoRR*, abs/1803.05494, 2018.

[11] H. Araujo and J. M. Dias. An introduction to the log-polar mapping [image sampling]. In *Proceedings II Workshop on Cybernetic Vision*, 1996.

[12] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman. Interactive object counting. In *ECCV*, 2014.

[13] C. Arteta, V. Lempitsky, and A. Zisserman. Counting in the wild. In *ECCV*, 2016.

[14] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE TPAMI*, 2017.

[15] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *IJCAI*, 1977.

[16] J. Bell and H. M. Dee. Aberystwyth leaf evaluation dataset, Nov. 2016.

[17] J. Bendig, K. Yu, H. Aasen, A. Bolten, S. Bennertz, J. Broscheit, M. L. Gnyp, and G. Bareth. Combining uav-based plant height from crop surface models, visible, and near infrared vegetation indices for biomass monitoring in barley. *International Journal of Applied Earth Observation and Geoinformation*, 2015.

[18] C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[19] L. Boominathan, S. S. S. Kruthiventi, and R. V. Babu. CrowdNet: A deep convolutional network for dense crowd counting. In *ACMM*, 2016.

[20] G. J. Brostow, J. Fauqueur, and R. Cipolla. Semantic object classes in video: A high-definition ground truth database. *PRL*, 2009.

[21] H. Caesar, J. R. R. Uijlings, and V. Ferrari. Coco-Stuff: Thing and stuff classes in context. *CoRR*, abs/1612.03716, 2016.

[22] K. Chen, C. C. Loy, S. Gong, and T. Xiang. Feature mining for localised crowd counting. In *BMVC*, 2012.

[23] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE TPAMI*, 2017.

[24] L. C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017.

[25] A. Coates, A. Ng, and H. Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 2011.

[26] J. P. Cohen, G. Boucher, C. A. Glastonbury, H. Z. Lo, and Y. Bengio. Count-ception: Counting by fully convolutional redundant counting. In *IEEE ICCVW*, 2017.

[27] R. Collobert, K. Kavukcuoglu, and C. Farabet. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*, 2011.

[28] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 1995.

[29] D. Costea, A. Marcu, M. Leordeanu, and E. Slusanschi. Creating roadmaps in aerial images with generative adversarial networks and smoothing-based optimization. In *IEEE ICCVW*, 2017.

[30] J. Dai, B. Bean, B. Brown, W. Bruening, J. Edwards, M. Flowers, R. Karow, C. Lee, G. Morgan, M. Ottman, J. Ransom, and J. Wiersma. Harvest index and straw yield of five classes of wheat. *Biomass and Bioenergy*, 2016.

[31] I. Demir, K. Koperski, D. Lindenbaum, G. Pang, J. Huang, S. Basu, F. Hughes, D. Tuia, and R. Raskar. DeepGlobe 2018: A Challenge to Parse the Earth through Satellite Images. *ArXiv e-prints*, May 2018.

[32] A. Dobrescu, M. V. Giuffrida, and S. A. Tsaftaris. Leveraging multiple datasets for deep leaf counting. In *IEEE ICCVW*, 2017.

[33] L. Fiaschi, U. Koethe, R. Nair, and F. A. Hamprecht. Learning to count with regression forest and structured labels. In *ICPR*, 2012.

[34] J. H. Friedman. Greedy function approximation: A gradient boosting machine. *Ann. Statist.*, 2001.

[35] R. T. Furbank and M. Tester. Phenomics – technologies to relieve the phenotyping bottleneck. *Trends in Plant Science*, 2011.

[36] R. Girshick. Fast R-CNN. In *IEEE ICCV*, 2015.

[37] M. V. Giuffrida, M. Minervini, and S. Tsaftaris. Learning to count leaves in rosette plants. In *Proceedings of the Computer Vision Problems in Plant Phenotyping (CVPPP)*, 2015.

[38] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AISTATS*, 2010.

[39] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. 2006.

[40] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*. Studies in Computational Intelligence. Springer, 2012.

[41] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *IEEE ICCV*, 2017.

[42] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE ICCV*, 2015.

[43] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE TPAMI*, 2015.

[44] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE CVPR*, 2016.

[45] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 1997.

[46] M. R. Hsieh, Y. L. Lin, and W. H. Hsu. Drone-based object counting by spatially regularized regional proposal network. In *IEEE ICCV*, 2017.

[47] G. Huang, Z. Liu, L. v. d. Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *IEEE CVPR*, 2017.

[48] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

[49] X. Jin, S. Liu, F. Baret, M. Hemerlé, and A. Comar. Estimates of plant density of wheat crops at emergence from very low altitude uav imagery. *Remote Sensing of Environment*, 2017.

[50] A. Josuttes, S. Aich, I. Stavness, C. Pozniak, and S. Shirtliffe. Utilizing deep learning to predict the number of spikes in wheat (triticum aestivum). In *Phenome 2018 Posters*, 2018.

[51] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[52] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*. 2011.

[53] P. Krähenbühl and V. Koltun. Parameter learning and convergent inference for dense random fields. In *ICML*, 2013.

[54] I. Krasin, T. Duerig, N. Alldrin, V. Ferrari, S. Abu-El-Haija, A. Kuznetsova, H. Rom, J. Uijlings, S. Popov, S. Kamali, M. Malloci, J. Pont-Tuset, A. Veit, S. Belongie, V. Gomes, A. Gupta, C. Sun, G. Chechik, D. Cai, Z. Feng, D. Narayanan, and K. Murphy. Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from https://storage.googleapis.com/openimages/web/index.html*, 2017.

[55] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*. 2012.

[56] G. V. Laurin, Q. Chen, J. A. Lindsell, D. A. Coomes, F. D. Frate, L. Guerriero, F. Pirotti, and R. Valentini. Above ground biomass estimation in an african tropical forest with lidar and hyperspectral data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 2014.

[57] G. V. Laurin, N. Puletti, Q. Chen, P. Corona, D. Papale, and R. Valentini. Above ground biomass and tree species richness estimation with airborne lidar in tropical ghana forests. *International Journal of Applied Earth Observation and Geoinformation*, 2016.

[58] K. Lawles, W. Raun, K. Desta, and K. Freeman. Effect of delayed emergence on corn grain yields. *Journal of Plant Nutrition*, 2012.

[59] A. Lehmussola, P. Ruusuvuori, J. Selinummi, H. Huttunen, and O. Yli-Harja. Computational framework for simulating fluorescence microscope images with cell populations. *IEEE TMI*, 2007.

[60] V. Lempitsky and A. Zisserman. Learning to count objects in images. In *NIPS*. 2010.

[61] G. Lin, A. Milan, C. Shen, and I. Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *IEEE CVPR*, 2017.

[62] M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.

[63] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014.

[64] S. Liu, F. Baret, B. Andrieu, P. Burger, and M. Hemmerlé. Estimation of wheat plant density at early stages using high resolution imagery. *Frontiers in Plant Science*, 2017.

[65] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, 2015.

[66] M. Marsden, K. McGuinness, S. Little, and N. E. O'Connor. Fully convolutional crowd counting on highly congested scenes. *CoRR*, abs/1612.00220, 2016.

[67] M. Minervini, M. M. Abdelsamea, and S. A. Tsaftaris. Image-based plant phenotyping with incremental learning and active contours. *Ecological Informatics*, 2014.

[68] M. Minervini, A. Fischbach, H. Scharr, and S. A. Tsaftaris. Finely-grained annotated datasets for image-based plant phenotyping. *PRL*, 2016.

[69] V. Mnih, N. Heess, A. Graves, and K. Kavukcuoglu. Recurrent models of visual attention. In *NIPS*. 2014.

[70] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In *ECCV*, 2016.

[71] G. Máttyus, W. Luo, and R. Urtasun. Deeproadmapper: Extracting road topology from aerial images. In *IEEE ICCV*, 2017.

[72] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE ICCV*, 2015.

[73] D. Oñoro-Rubio and R. J. López-Sastre. Towards perspective-free object counting with deep learning. In *ECCV*, 2016.

[74] J.-M. Pape and C. Klukas. 3-d histogram-based segmentation and leaf detection for rosette plants. In *ECCV Workshop*, 2015.

[75] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun. Large kernel matters – improve semantic segmentation by global convolutional network. In *IEEE CVPR*, 2017.

[76] PKLot - A Robust Dataset for Parking Lot Classification. *Expert Syst. Appl.*, 2015.

[77] M. P. Pound, J. A. Atkinson, D. M. Wells, T. P. Pridmore, and A. P. French. Deep learning for multi-task plant phenotyping. In *IEEE ICCVW*, 2017.

[78] PyTorch. http://pytorch.org/.

[79] X. Quan, B. He, M. Yebra, C. Yin, Z. Liao, X. Zhang, and X. Li. A radiative transfer model-based method for the estimation of grassland aboveground biomass. *International Journal of Applied Earth Observation and Geoinformation*, 2017.

[80] B. Reddersen, T. Fricke, and M. Wachendorf. A multi-sensor approach for predicting biomass of extensively managed grassland. *Computers and Electronics in Agriculture*, 2014.

[81] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *IEEE CVPR*, 2016.

[82] M. Ren and R. S. Zemel. End-to-end instance segmentation with recurrent attention. In *IEEE CVPR*, 2017.

[83] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*. 2015.

[84] B. Romera-Paredes and P. Torr. Recurrent instance segmentation. In *ECCV*, 2016.

[85] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[86] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. F.-F. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015.

[87] D. B. Sam, S. Surya, and R. V. Babu. Switching convolutional neural network for crowd counting. In *IEEE CVPR*, 2017.

[88] H. Scharr, M. Minervini, A. Fischbach, and S. A. Tsaftaris. Annotated Image Datasets of Rosette Plants. Technical Report FZJ-2014-03837, 2014.

[89] H. Scharr, M. Minervini, A. P. French, C. Klukas, D. M. Kramer, X. Liu, I. Luengo, J.-M. Pape, G. Polder, D. Vukadinovic, X. Yin, and S. A. Tsaftaris. Leaf segmentation in plant phenotyping: a collation study. *MVAP*, 2016.

[90] M. Schirrmann, A. Hamdorf, A. Garz, A. Ustyuzhanin, and K.-H. Dammer. Estimating wheat biomass by combining image clustering with crop height. *Computers and Electronics in Agriculture*, 2016.

[91] S. Seguí, O. Pujol, and J. Vitrià. Learning to count with deep object features. In *IEEE CVPRW*, 2015.

[92] P. J. SELLERS. Canopy reflectance, photosynthesis and transpiration. *International Journal of Remote Sensing*, 1985.

[93] C. Shang, H. Ai, and B. Bai. End-to-end crowd counting via joint learning local and global count. In *IEEE ICIP*, 2016.

[94] W. Shi, J. Caballero, F. Huszár, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE CVPR*, 2016.

[95] D. Shrestha and B. Steward. Shape and size analysis of corn plant canopies for plant population and spacing sensing. *Applied Eng. in Agric*, 2005.

[96] D. Shrestha, B. Steward, and S. Birrell. Video processing for early stage maize plant detection. *Biosystems engineering*, 2004.

[97] D. S. Shrestha and B. L. Steward. Automatic corn plant population measurement using machine vision, paper number 011067. In *2001 ASAE Annual Meeting*, 2001.

[98] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[99] V. A. Sindagi and V. M. Patel. CNN-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *IEEE AVSS*, 2017.

[100] V. A. Sindagi and V. M. Patel. A survey of recent advances in cnn-based single image crowd counting and density estimation. *PRL*, 2017.

[101] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *IEEE CVPR*, 2015.

[102] J. M. Soriano, M. Malosetti, M. Roselló, M. E. Sorrells, and C. Royo. Dissecting the old mediterranean durum wheat genetic architecture for phenology, biomass and yield formation by association mapping and qtl meta-analysis. *PLOS ONE*, 2017.

[103] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE CVPR*, 2015.

[104] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *IEEE CVPR*, 2016.

[105] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. In *IEEE CVPR*, 2015.

[106] J. R. Ubbens and I. Stavness. Deep plant phenomics: A deep learning platform for complex plant phenotyping tasks. *Frontiers in Plant Science*, 2017.

[107] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE TPAMI*, 1991.

[108] E. Walach and L. Wolf. Learning to count with cnn boosting. In *ECCV*, 2016.

[109] C. Wang, H. Zhang, L. Yang, S. Liu, and X. Cao. Deep people counting in extremely dense crowds. In *ACMM*, 2015.

[110] Z. Wojna, V. Ferrari, S. Guadarrama, N. Silberman, L. chieh Chen, A. Fathi, and J. Uijlings. The devil is in the decoders. In *BMVC*, 2017.

[111] W. Xie, J. A. Noble, and A. Zisserman. Microscopy cell counting and detection with fully convolutional regression networks. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 2016.

[112] Y. Xie, F. Xing, H. Su, and L. Yang. Beyond classification: Structured regression for robust cell detection using convolutional neural network. In *MICCAI*, 2015.

[113] X. Yin, X. Liu, J. Chen, and D. M. Kramer. Multi-leaf alignment from fluorescence plant images. In *IEEE WACV*, 2014.

[114] X. Yin, X. Liu, J. Chen, and D. M. Kramer. Multi-leaf tracking from fluorescence plant videos. In *IEEE ICIP*, 2014.

[115] X. Yin, X. Liu, J. Chen, and D. M. Kramer. Joint multi-leaf segmentation, alignment and tracking from fluorescence plant videos. *CoRR*, abs/1505.00353, 2015.

[116] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.

[117] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. In *IEEE CVPR*, 2015.

[118] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma. Single-image crowd counting via multi-column convolutional neural network. In *IEEE CVPR*, 2016.

[119] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *IEEE CVPR*, 2017.

[120] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. S. Torr. Conditional random fields as recurrent neural networks. In *IEEE ICCV*, 2015.

[121] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene CNNs. *CoRR*, abs/1412.6856, 2014.

[122] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. In *IEEE CVPR*, 2016.

[123] L. Zitnick and P. Dollar. Edge boxes: Locating object proposals from edges. In *ECCV*, 2014.